UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

TRABAJO FIN DE MÁSTER

Gestión de Trabajos en Movilidad (GTM) v2.0 (INF.TFM.13.14.09)

Alba Calvo Ruiz

Septiembre 2014

Gestión de Trabajos en Movilidad (GTM) v2.0

AUTOR: Alba Calvo Ruiz
TUTOR: Álvaro Ortigosa Juárez

Dpto. de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Septiembre de 2014

RESUMEN

El proyecto GTM (Gestión de Trabajos en Movilidad) v2.0 es una continuación de mi Trabajo de Fin de Grado "Alba Calvo Ruiz. Gestión de Trabajos en Movilidad (GTM). (COSITI_120)", consistente en una aplicación de gestión para empresas SAT (Servicio de Atención Técnica) con soporte de movilidad.

GTM v2.0 gestiona el ciclo completo de vida de un aviso, incluyendo aspectos como la recepción, asignación y envío a técnicos, cumplimentación del trabajo realizado, etc.

La aplicación proporciona servicios de movilidad. Los técnicos reciben automáticamente los avisos en sus terminales móviles, desde donde graban el trabajo realizado a la finalización del aviso. Estos servicios están disponibles tanto en modo conectado como desconectado, pudiendo los técnicos cumplimentar los avisos aunque carezcan de conexión de red, produciéndose la sincronización con la base de datos de la aplicación corporativa más adelante cuando el técnico disponga de conectividad.

GTM v2.0 proporciona también subida de fotos asociadas a avisos, así como tickets (justificantes) firmados por el cliente en la pantalla del terminal en movilidad, quedando ambos tipos de fichero almacenados en el repositorio de la gestión documental de la aplicación.

Además, GTM v2.0 soporta la visualización de mapas de ubicación de clientes, asignación de avisos y control de rutas de técnicos.

La implementación constituye un sistema integrado por una aplicación corporativa de gestión de avisos y presentación de mapas de clientes, avisos y rutas de técnicos, y de una aplicación para terminales móviles de tracking GPS y de recepción y cumplimentación de trabajos por parte de los técnicos.

Palabras clave: movilidad, geolocalización, tracking GPS, terminal móvil, gestión de avisos.

ABSTRACT

The GTM (Gestión de Trabajos en Movilidad) v2.0 project is a continuation of my final year dissertation "Alba Calvo Ruiz. Gestión de Trabajos en Movilidad (GTM). (COSITI_120)", consisting in a management application for companies providing mobile Technical Assistance Service (TAS).

GTM v2.0 manages the complete cycle of a notice, including aspects such as reception, assignment and dispatch of technicians, fulfilment of a task, etc.

The application provides mobility services. Technicians receive tasks automatically on their mobile terminals, where they can also save the task accomplished once finished. These services are accessible both in online and offline mode, being the technicians able to fulfil the tasks even when network connection is not available. The synchronization with the corporate database takes place later when connectivity is available.

GTM v2.0 also provides uploading of photos relating to tasks, as well as tickets signed by clients in the screen of the mobile terminal. Both types of files are saved to the repository of the application document management.

In addition, GTM v2.0 provides visual maps with the location of clients, task assignment and routes of technicians.

The implementation forms an integrated system made up of a corporate application of task management and mapping representation of clients, tasks and routes of technicians, and an application for mobile terminals with GPS tracking and reception and fulfilment of the tasks carried out by technicians.

Keywords: mobility, georeference, GPS tracking, mobile terminal, notice management service.

Agradecimientos

Me gustaría dar las gracias a mi familia por su apoyo incondicional en todo momento.

Mención especial merecen mis padres Miguel y Paz, el primero por ser el "culpable" de que decidiera dedicarme al mundillo de la informática y por ayudarme siempre que lo he necesitado, y la segunda por haberme cuidado y aconsejado tan bien durante todo este tiempo. Sin ellos no habría llegado hasta aquí.

No me olvido de mi hermano pequeño Miguel, que ahora tiene que enfrentarse a la etapa universitaria que yo ya voy a dejar atrás, aunque estoy segura de que lo hará muy bien.

También les agradezco a los profesores de la EPS el esfuerzo que han dedicado a formarme, especialmente a mi tutor Álvaro Ortigosa, que me ha prestado ayuda con mi proyecto siempre que lo he necesitado y que se ha encargado de supervisarlo y corregirlo.

INDICE DE CONTENIDOS

Glosario 9
1 Introducción 11
1.1 Antecedentes 11
1.2 Objetivos 11
1.3 Motivación 12
1.4 Organización de la memoria 14
2 Estado del arte 15
2.1 Introducción 15
2.2 Aplicación cliente de escritorio 15
2.2.1 Java 15
2.2.1.1 Ventajas 15
2.2.1.2 Desventajas 15
2.2.2 .Net 16
2.2.2.1 Ventajas 16
2.2.2.2 Desventajas 16
2.3 Plataformas de referencia para terminales en movilidad. 16
2.3.1 Windows phone 16
2.3.1.1 Ventajas 16
2.3.1.2 Desventajas 16
2.3.2 Blackberry OS 16
2.3.2.1 Ventajas 16
2.3.2.2 Desventajas 17
2.3.3 iOS 17
2.3.3.1 Ventajas 17
2.3.3.2 Desventajas 17
2.3.4 Android 17
2.3.4.1 Ventajas 17
2.3.4.2 Desventajas 18
2.4 Servicios Web 18
2.4.1 SOAP 18
2.4.1.1 Ventajas 18
2.4.1.2 Desventajas. 18
2.4.2 REST 19
2.4.2.1 Ventajas 19
2.4.2.2 Desventajas. 19
2.5 Representación cartográfica 19
2.5.1 Google Maps 19
2.5.1.1 Ventajas 19
2.5.1.2 Desventajas 20
2.5.2 MapPoint 20
2.5.2.1 Ventajas 20
2.5.2.2 Desventajas 20

2.6 Base de Datos. 21
2.6.1 SQLServer 21
2.6.1.1 Ventajas 21
2.6.1.2 Desventajas 21
2.6.2 MySQL. 21
2.6.2.1 Ventajas 21
2.6.2.2 Desventajas 21
2.6.3 PostgreSQL 22
2.6.3.1 Ventajas 22
2.6.3.2 Desventajas 22
3 Diseño 23
3.1 Arquitectura 23
3.2 Esquema de base de datos 25
3.3 Interfaz de usuario 27
3.3.1 Aplicación cliente de gestión 27
3.3.1.1 Empleado 27
3.3.1.2 Administrador 28
3.3.1.2.1 Clientes – Avisos – Documentos 29
3.3.1.2.2 Avisos pendientes 34
3.3.1.2.3 Gestión documental 36
3.3.1.2.4 Rutas de técnicos 37
3.3.1.2.5 Empleados y Técnicos 39
3.3.1.2.6 Usuarios 40
3.3.1.2.7 Mis datos 41
3.3.1.2.8 Logout 41
3.3.2 Aplicación para terminales en movilidad 42
3.3.2.1 Tracking GPS 43
3.3.2.2 Gestión de avisos 45
3.3.2.2.1 Avisos Nuevos 46
3.3.2.2.2 Avisos Pendientes 47
3.3.2.2.3 Historial de avisos realizados el último mes 49
3.3.2.3 Logout 50
3.4 Ejemplo de flujo de la aplicación 51
3.5 Componentes del proyecto. Análisis de tecnologías 56
3.5.1 Base de datos SQLServer 57
3.5.2 Aplicación cliente .Net con formularios Windows Forms 57
3.5.3 Aplicación cliente de gestión de avisos y tracking Android 59
3.5.4 API Web REST en PHP 59
3.5.5 Representación cartográfica con MapPoint 60
4 Implementación 61
4.1 Base de datos 61
4.1.1 Tecnologías empleadas 61

4.2 Aplicación cliente de gestión y representación cartográfica 61
4.2.1 Tecnologías empleadas 61
4.2.2 Estructura 61
4.2.2.1 ModeloDTO y Negocio 62
4.2.2.2 MovilidadPresentacion 65
4.2.2.3 Diagrama UML 65
4.2.2.4 Diagrama de secuencia 67
4.3 API Web 68
4.3.1 Tecnologías empleadas 68
4.3.2 Estructura 68
4.3.2.1 Conexion.class.php 68
4.3.2.2 restApi.php 70
4.4 Aplicación de geolocalización y gestión de avisos en terminales en movilidad 72
4.4.1 Tecnologías empleadas 72
4.4.2 Estructura 72
4.4.3 Sincronización de base de datos y repositorio de ficheros locales con los alojados en el servidor Web 72
4.4.4 Tarea asíncrona de login 73
4.4.5 Servicio de tracking 74
4.4.6 Interfaz de usuario 75
4.4.7 Diagramas 75
4.4.7.1 Diagrama UML 75
4.4.7.2 Diagramas de secuencia 76
5 Integración, pruebas y resultados 77
5.1 Pruebas modulares 77
5.1.1 Pruebas de aplicación cliente de gestión 77
5.1.1.1 Entorno de prueba 77
5.1.1.2 Pruebas realizadas 77
5.1.1.2.1 Pruebas de correcto almacenamiento en la base de datos 77
5.1.1.2.2 Pruebas de interfaz 84
5.1.1.2.3 Pruebas de presentación de mapas 85
5.1.1.2.4 Pruebas de visualización de documentos 87
5.1.2 Pruebas API Web 90
5.1.2.1 Diseño de la prueba 90
5.1.2.2 Análisis de resultados 90
5.1.3 Pruebas de aplicación móvil de gestión de avisos y tracking GPS 92
5.1.3.1 Pruebas realizadas 92
5.1.3.1.1 Pruebas de realización de login contra la API Web 92
5.1.3.1.2 Pruebas de servicio de tracking GPS 93
5.1.3.1.3 Pruebas de gestión de avisos 94
5.2 Pruebas de integración 98
5.2.1 Diseño de la prueba 98
5.2.2 Análisis de resultados 98

6 Conclusiones y trabajo futuro 99
6.1 Conclusiones 99
6.2 Trabajo futuro 100
Bibliografía 103

Anexos I
A Manual de instalación I
1 Introducción iii
2 Requisitos software iii
3 Requisitos de acceso a recursos del servidor iii
4 Posibles escenarios de instalación iii
4.1 Casuística de escenarios iii
4.2 Paquetes a instalar en cada caso. iv
5 Descripción de la instalación de paquetes iv
5.1 Instalación en PCs iv
5.2 Instalación en dispositivos móviles con Android v

INDICE DE FIGURAS

FIGURA 1. ESQUEMA GTM V1.0. SUBSISTEMA DE GESTIÓN DE CLIENTES Y AVISOS. 23
FIGURA 2. ESQUEMA GTM V1.0. SUBSISTEMA DE TRACKING GPS Y REPRESENTACIÓN CARTOGRÁFICA. 23
FIGURA 3. ESQUEMA DE NUEVA FUNCIONALIDAD GTM V2.0. 24
FIGURA 4. ESQUEMA DE LA BASE DE DATOS. 26
FIGURA 5. APLICACIÓN CLIENTE .NET: PANTALLA DE LOGIN. 27
FIGURA 6. APLICACIÓN CLIENTE .NET: MENÚ DE EMPLEADO. 27
FIGURA 7. APLICACIÓN CLIENTE .NET: MÁQUINA DE ESTADOS DE EMPLEADO. 28
FIGURA 8. APLICACIÓN CLIENTE .NET: MENÚ DE ADMINISTRADOR. 28
FIGURA 9. APLICACIÓN CLIENTE .NET: MÁQUINA DE ESTADOS DE ADMINISTRADOR. 29
FIGURA 10. APLICACIÓN CLIENTE .NET: VENTANA DE CLIENTES-AVISOS-DOCUMENTOS. 29
FIGURA 11. APLICACIÓN CLIENTE .NET: ALTAS, BAJAS, MODIFICACIONES. 30
FIGURA 12. APLICACIÓN CLIENTE .NET: ALTA DE AVISO. 30
FIGURA 13. APLICACIÓN CLIENTE .NET: REJILLA DE AVISOS CON CAMBIOS REFLEJADOS. 31
FIGURA 14. APLICACIÓN CLIENTE .NET: MENSAJE DE NOTIFICACIÓN DE RECEPCIÓN DE AVISO. 31
FIGURA 15. APLICACIÓN CLIENTE .NET: VER MAPA DE CLIENTES. 31
FIGURA 16. APLICACIÓN CLIENTE .NET: MAPA DE CLIENTES DE LA APLICACIÓN. 32
FIGURA 17. APLICACIÓN CLIENTE .NET: INFORMACIÓN DE CLIENTE. 32
FIGURA 18. APLICACIÓN CLIENTE .NET: DOCUMENTOS ASOCIADOS A UN AVISO. 32
FIGURA 19. APLICACIÓN CLIENTE .NET: VISUALIZACIÓN DE UN DOCUMENTO. 33
FIGURA 20. APLICACIÓN CLIENTE .NET: DOCUMENTO (FOTO Y TICKET, RESPECTIVAMENTE) 33
FIGURA 21. APLICACIÓN CLIENTE .NET: DOCUMENTOS ASOCIADOS A UN CLIENTE 33
FIGURA 22. APLICACIÓN CLIENTE .NET: ALTA DE AVISO SIN TÉCNICO ASOCIADO. 34
FIGURA 23. APLICACIÓN CLIENTE .NET: PRESENTACIÓN DE AVISOS SIN TÉCNICO ASOCIADO EN ROJO EN LA REJILLA DE DETALLE DEL FORMULARIO CLIENTES-AVISOS-DOCUMENTOS. 34
FIGURA 24. APLICACIÓN CLIENTE .NET: REJILLA DE AVISOS PENDIENTES. 35
FIGURA 25. APLICACIÓN CLIENTE .NET: MAPA DE AVISOS PENDIENTES DE REALIZACIÓN (MARCADOR AZUL PARA ASIGNADOS A TÉCNICO Y GRIS PARA NO ASIGNADOS) 35
FIGURA 26. APLICACIÓN CLIENTE .NET: MAPA DE AVISOS PENDIENTES FILTRADOS POR TÉCNICO ASIGNADO. 35
FIGURA 27. APLICACIÓN CLIENTE .NET: ASIGNACIÓN DE AVISO PENDIENTE A TÉCNICO MEDIANTE CLICK EN EL MARCADOR GRIS. 36

FIGURA 28. APLICACIÓN CLIENTE .NET: GRABACIÓN DE AVISO PENDIENTE ASOCIADO A TÉCNICO Y PRESENTACIÓN EN EL MAPA (MARCADOR AZUL)..... 36
FIGURA 29. APLICACIÓN CLIENTE .NET: REJILLA DE GESTIÓN DOCUMENTAL. 37
FIGURA 30. APLICACIÓN CLIENTE .NET: PRESENTACIÓN DE RUTA DE TÉCNICO DADA UNA FECHA.37
FIGURA 31. APLICACIÓN CLIENTE .NET: MAPA DE RUTA DE TÉCNICO. MARCADORES DE PARADA.37
FIGURA 32. APLICACIÓN CLIENTE .NET: MAPA DE RUTA DE TÉCNICO. MARCADORES DE AVISOS. 37
FIGURA 33. APLICACIÓN CLIENTE .NET: MAPA DE RUTA DE TÉCNICO. DETECCIÓN DE PARADAS "SOSPECHOSAS"..... 38
FIGURA 34. APLICACIÓN CLIENTE .NET: MAPA DE RUTA DE TÉCNICO. FICHA DE INFORMACIÓN DE PARADA..... 38
FIGURA 35. APLICACIÓN CLIENTE .NET: MAPA DE RUTA DE TÉCNICO. FICHA DE INFORMACIÓN DE AVISO (CON OPCIÓN DE VISUALIZACIÓN DE TICKET FIRMADO POR EL CLIENTE)..... 38
FIGURA 36. APLICACIÓN CLIENTE .NET: REJILLA DE EMPLEADOS. 39
FIGURA 37. APLICACIÓN CLIENTE .NET: ALTA DE EMPLEADO. 39
FIGURA 38. APLICACIÓN CLIENTE .NET: REJILLA DE EMPLEADOS CON CAMBIOS REFLEJADOS. 40
FIGURA 39. APLICACIÓN CLIENTE .NET: REJILLA DE USUARIO CON CAMBIOS REFLEJADOS. 40
FIGURA 40. APLICACIÓN CLIENTE .NET: MODIFICACIÓN DE USUARIO. 41
FIGURA 41. APLICACIÓN CLIENTE .NET: MODIFICACIÓN DE DATOS DE USUARIO. 41
FIGURA 42. APLICACIÓN CLIENTE ANDROID: MÁQUINA DE ESTADOS..... 42
FIGURA 43. APLICACIÓN CLIENTE ANDROID: LOGIN. 42
FIGURA 44. APLICACIÓN CLIENTE ANDROID: MENÚ PRINCIPAL..... 43
FIGURA 45. APLICACIÓN CLIENTE ANDROID: MENÚ DE TRACKING GPS. 43
FIGURA 46. APLICACIÓN CLIENTE ANDROID: DETECCIÓN DE HABILITACIÓN DEL GPS DEL DISPOSITIVO..... 44
FIGURA 47. APLICACIÓN CLIENTE ANDROID: INICIO DEL SERVICIO DE TRACKING GPS. 44
FIGURA 48. APLICACIÓN CLIENTE ANDROID: FINALIZACIÓN DEL SERVICIO DE TRACKING GPS. ... 45
FIGURA 49. APLICACIÓN CLIENTE ANDROID: MENÚ DE GESTIÓN DE AVISOS..... 45
FIGURA 50. APLICACIÓN CLIENTE ANDROID: MENÚ DE AVISOS NUEVOS Y DETALLE DE LOS DATOS DE UN AVISO NUEVO. 46
FIGURA 51. APLICACIÓN CLIENTE ANDROID: SINCRONIZACIÓN DEL AVISO CON LA BASE DE DATOS WEB TRAS MARCARLO COMO LEÍDO. 46
FIGURA 52. APLICACIÓN CLIENTE ANDROID: MENÚ DE AVISOS PENDIENTES Y DETALLE DE UN AVISO PENDIENTE. 47
FIGURA 53. APLICACIÓN CLIENTE ANDROID: ADICIÓN DE IMÁGENES AL CUMPLIMENTAR UN AVISO PENDIENTE..... 47
FIGURA 54. APLICACIÓN CLIENTE ANDROID: VISUALIZACIÓN DE IMÁGENES AÑADIDAS (CON POSIBILIDAD DE ELIMINACIÓN) AL CUMPLIMENTAR UN AVISO. 48
FIGURA 55. APLICACIÓN CLIENTE ANDROID: GENERACIÓN DE UN TICKET FIRMADO POR EL CLIENTE AL GRABAR EL AVISO CUMPLIMENTADO. 48
FIGURA 56. APLICACIÓN CLIENTE ANDROID: MENÚ DE HISTORIAL DE AVISOS REALIZADOS EN EL ÚLTIMO MES..... 49
FIGURA 57. APLICACIÓN CLIENTE ANDROID: DETALLE DE UN AVISO DEL HISTORIAL, INCLUYENDO VISUALIZACIÓN DE IMÁGENES ASOCIADAS Y DE TICKET FIRMADO POR EL CLIENTE (EN CASO DE TENERLOS)..... 49
FIGURA 58. APLICACIÓN CLIENTE ANDROID: GENERACIÓN DE TICKET FIRMADO POR EL CLIENTE DESDE EL DETALLE DE UN AVISO DEL HISTORIAL..... 50
FIGURA 59. APLICACIÓN CLIENTE .NET: ALTA DE AVISO SIN TÉCNICO ASOCIADO. 51
FIGURA 60. APLICACIÓN CLIENTE .NET: AVISOS SIN TÉCNICO ASOCIADO INDICADOS EN ROJO EN LA REJILLA CLIENTES-AVISOS-DOCUMENTOS. 51
FIGURA 61. APLICACIÓN CLIENTE .NET: REJILLA DE AVISOS PENDIENTES..... 51
FIGURA 62. APLICACIÓN CLIENTE .NET: MAPA DE AVISOS PENDIENTES DE REALIZACIÓN (MARCADOR AZUL PARA ASIGNADOS A TÉCNICO Y GRIS PARA NO ASIGNADOS)..... 52
FIGURA 63. APLICACIÓN CLIENTE .NET: MAPA DE AVISOS PENDIENTES FILTRADOS POR TÉCNICO ASIGNADO..... 52

FIGURA 64. APLICACIÓN CLIENTE .NET: ASIGNACIÓN DE AVISO PENDIENTE A TÉCNICO MEDIANTE CLICK EN EL MARCADOR GRIS. 52
FIGURA 65. APLICACIÓN CLIENTE .NET: GRABACIÓN DE AVISO PENDIENTE ASOCIADO A TÉCNICO Y PRESENTACIÓN EN EL MAPA (MARCADOR AZUL)..... 53
FIGURA 66. MENSAJE DE NOTIFICACIÓN DE RECEPCIÓN DE AVISO. 53
FIGURA 67. APLICACIÓN CLIENTE ANDROID: AUTENTICACIÓN DE TÉCNICO Y ACCESO AL MENÚ DE GESTIÓN DE AVISOS. 53
FIGURA 68. APLICACIÓN CLIENTE ANDROID: MARCADO DE AVISO NUEVO COMO LEÍDO. 54
FIGURA 69. APLICACIÓN CLIENTE ANDROID: CUMPLIMENTACIÓN DE AVISO PENDIENTE. 54
FIGURA 70. APLICACIÓN CLIENTE ANDROID: CUMPLIMENTACIÓN DE AVISO PENDIENTE. ADICIÓN DE IMÁGENES. 55
FIGURA 71. APLICACIÓN CLIENTE ANDROID: GRABACIÓN DE AVISO CON OPCIÓN DE GENERACIÓN DE TICKET FIRMADO POR EL CLIENTE HABILITADA. 55
FIGURA 72. APLICACIÓN CLIENTE ANDROID: MENÚ DE HISTORIAL DE AVISOS DEL ÚLTIMO MES Y DETALLE DEL AVISO REALIZADO. 56
FIGURA 73. APLICACIÓN CLIENTE ANDROID: VISUALIZACIÓN DE IMÁGENES Y TICKET ASOCIADOS AL AVISO. 56
FIGURA 74. ESQUEMA GTM V2.0..... 57
FIGURA 75. ARQUITECTURA DE UNA APLICACIÓN WEB. 58
FIGURA 76. ARQUITECTURA DE UNA APLICACIÓN CLIENTE..... 58
FIGURA 77: CAPAS DE LA APLICACIÓN CLIENTE. 61
FIGURA 78. MOVILIDADPRESENTACION: DIAGRAMA UML. 65
FIGURA 79. APLICACIÓN CLIENTE .NET: FORMULARIO MDI 1. 66
FIGURA 80. APLICACIÓN CLIENTE .NET: FORMULARIO MDI 2. 66
FIGURA 81. APLICACIÓN CLIENTE .NET: DIAGRAMA DE SECUENCIA..... 67
FIGURA 82. APLICACIÓN CLIENTE ANDROID: SINCRONIZACIÓN. PATRÓN SYNCADAPTER..... 72
FIGURA 83. APLICACIÓN CLIENTE ANDROID: DIAGRAMA UML. 75
FIGURA 84. APLICACIÓN CLIENTE ANDROID: DIAGRAMA DE SECUENCIA DE TRACKING. 76
FIGURA 85. APLICACIÓN CLIENTE ANDROID: DIAGRAMA DE SECUENCIA DE GESTIÓN DE AVISOS. 76
FIGURA 86. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE .NET: REJILLA DE EMPLEADOS. 78
FIGURA 87. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE .NET: CONSULTA DE EMPLEADOS. 78
FIGURA 88. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE .NET: CONSULTA DE USUARIOS..... 78
FIGURA 89. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE .NET: ALTA DE EMPLEADO.79
FIGURA 90. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE .NET: CONSULTA DE EMPLEADOS CON CAMBIOS REFLEJADOS. 79
FIGURA 91. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE: CONSULTA DE USUARIOS CON CAMBIOS REFLEJADOS..... 79
FIGURA 92. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE .NET: REJILLA DE EMPLEADOS CON CAMBIOS REFLEJADOS. 80
FIGURA 93. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE .NET: REJILLA DE USUARIOS CON CAMBIOS REFLEJADOS..... 80
FIGURA 94. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE .NET: MODIFICACIÓN DE EMPLEADO. 80
FIGURA 95. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE .NET: CONSULTA DE EMPLEADOS CON CAMBIOS REFLEJADOS 2. 81
FIGURA 96. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE .NET: CONSULTA DE USUARIOS CON CAMBIOS REFLEJADOS 2..... 81
FIGURA 97. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE .NET: REJILLA DE EMPLEADOS CON CAMBIOS REFLEJADOS 2. 81
FIGURA 98. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE .NET: REJILLA DE USUARIOS CON CAMBIOS REFLEJADOS 2..... 82

FIGURA 99. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE .NET: ELIMINACIÓN DE EMPLEADO. 82
FIGURA 100. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE .NET: CONSULTA DE EMPLEADOS CON CAMBIOS REFLEJADOS 3. 82
FIGURA 101. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE .NET: CONSULTA DE USUARIOS CON CAMBIOS REFLEJADOS 3. 83
FIGURA 102. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE .NET: REJILLA DE EMPLEADOS CON CAMBIOS REFLEJADOS 3. 83
FIGURA 103. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE .NET: REJILLA DE USUARIOS CON CAMBIOS REFLEJADOS 3. 83
FIGURA 104. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE .NET: PRUEBA DE CAMPOS SIN RELLENAR. 84
FIGURA 105. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE .NET: PRUEBA DE CAMPOS DEMASIADO LARGOS. 84
FIGURA 106. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE .NET: PRUEBA DE PRESENTACIÓN DE MAPAS. CONSULTA DE DIRECCIONES DE CLIENTES. 85
FIGURA 107. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE .NET: PRUEBA DE PRESENTACIÓN DE MAPAS. MAPA DE LOCALIZACIÓN DE CLIENTES. 85
FIGURA 108. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE .NET: PRUEBA DE PRESENTACIÓN DE MAPAS. FICHA DE INFORMACIÓN DE CLIENTE. 86
FIGURA 109. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE .NET: PRUEBA DE PRESENTACIÓN DE MAPAS. FICHA DE INFORMACIÓN DE AVISO. 86
FIGURA 110. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE .NET: PRUEBA DE PRESENTACIÓN DE MAPAS. FICHA DE INFORMACIÓN DE PARADA. 87
FIGURA 111. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE .NET: PRUEBA DE VISUALIZACIÓN DE DOCUMENTOS. DOCUMENTOS DEL REPOSITORIO WEB. 87
FIGURA 112. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE .NET: PRUEBA DE VISUALIZACIÓN DE DOCUMENTOS. VISUALIZACIÓN DE DOCUMENTO DEL REPOSITORIO WEB DESDE EL NAVEGADOR. 88
FIGURA 113. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE .NET: PRUEBA DE VISUALIZACIÓN DE DOCUMENTOS. DOCUMENTOS ASOCIADOS A AVISO. 88
FIGURA 114. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE .NET: PRUEBA DE VISUALIZACIÓN DE DOCUMENTOS. VISUALIZACIÓN DE DOCUMENTO DESDE LA APLICACIÓN CLIENTE .NET. 88
FIGURA 115. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE .NET: PRUEBA DE VISUALIZACIÓN DE DOCUMENTOS. DOCUMENTOS ASOCIADOS A CLIENTE. 89
FIGURA 116. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE .NET: PRUEBA DE VISUALIZACIÓN DE DOCUMENTOS. VISUALIZACIÓN DE DOCUMENTO DESDE LA APLICACIÓN CLIENTE .NET 2. 89
FIGURA 117. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE .NET: PRUEBA DE VISUALIZACIÓN DE DOCUMENTOS. DOCUMENTO JUSTIFICANTE FIRMADO POR EL CLIENTE... 89
FIGURA 118. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN API WEB: PRUEBA DE CONSULTA A LA BASE DE DATOS Y OBTENCIÓN DE OBJETOS JSON POR EL MÉTODO GET. 90
FIGURA 119. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN API WEB: PRUEBA DE INSERCIÓN EN BASE DE DATOS MEDIANTE EL ENVÍO DE UN OBJETO JSON POR EL MÉTODO POST. 91
FIGURA 120. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN API WEB: PRUEBA DE CORRECTA INSERCIÓN EN BASE DE DATOS DE LA COORDENADA ENVIADA EN FORMA DE OBJETO JSON EN EL PASO ANTERIOR. 91
FIGURA 121. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE ANDROID: PRUEBA DE AUTENTICACIÓN CONTRA LA API WEB. 92
FIGURA 122. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE ANDROID: MENÚ DE LA APLICACIÓN. 92
FIGURA 123. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE ANDROID: PRUEBA DE INICIO Y DETENCIÓN DEL SERVICIO DE TRACKING GPS. 93

FIGURA 124. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE ANDROID: CONSULTA DE LAS COORDENADAS ALMACENADAS EN BASE DE DATOS POR EL SERVICIO DE TRACKING GPS. 93
FIGURA 125. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE ANDROID: PRUEBA DE GESTIÓN DE AVISOS. CONSULTA DE AVISOS ASOCIADOS A TÉCNICO. 94
FIGURA 126. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE ANDROID: PRUEBA DE GESTIÓN DE AVISOS. MENÚ DE AVISOS NUEVOS. 94
FIGURA 127. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE ANDROID: PRUEBA DE GESTIÓN DE AVISOS. PRUEBA DE MARCADO DE AVISO COMO LEÍDO EN LA APLICACIÓN CLIENTE ANDROID. 95
FIGURA 128. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE ANDROID: PRUEBA DE GESTIÓN DE AVISOS. PRUEBA DE MARCADO DE AVISO COMO LEÍDO MEDIANTE CONSULTA A BASE DE DATOS. 95
FIGURA 129. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE ANDROID: PRUEBA DE GESTIÓN DE AVISOS. PRUEBA DE CUMPLIMENTACIÓN DE AVISO. CONSULTA INICIAL A LA BASE DE DATOS. 95
FIGURA 130. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE ANDROID: PRUEBA DE GESTIÓN DE AVISOS. PRUEBA CUMPLIMENTACIÓN DE AVISO EN LA APLICACIÓN CLIENTE ANDROID. 96
FIGURA 131. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE ANDROID: PRUEBA DE GESTIÓN DE AVISOS. PRUEBA DE CUMPLIMENTACIÓN DE AVISO. INTRODUCCIÓN DE DATOS RELATIVOS AL AVISO (HORA DE INICIO, TRABAJO REALIZADO, ESTADO), ADICIÓN DE FOTOS Y GENERACIÓN DE TICKET FIRMADO POR EL CLIENTE. 96
FIGURA 132. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE ANDROID: PRUEBA DE GESTIÓN DE AVISOS. ALMACENAMIENTO DE FOTOS Y TICKET FIRMADO POR EL CLIENTE ASOCIADOS A UN AVISO EN EL REPOSITORIO DE FICHEROS LOCAL DEL TELÉFONO. 97
FIGURA 133. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE ANDROID: PRUEBA DE GESTIÓN DE AVISOS. ALMACENAMIENTO DE FOTOS ASOCIADAS A UN AVISO EN EL REPOSITORIO WEB 97
FIGURA 134. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE ANDROID: PRUEBA DE GESTIÓN DE AVISOS. ALMACENAMIENTO DE TICKET GENERADO PARA UN AVISO EN EL REPOSITORIO WEB. 97
FIGURA 135. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE ANDROID: PRUEBA DE GESTIÓN DE AVISOS. ALMACENAMIENTO DE FOTOS Y TICKET EN LA BASE DE DATOS 98
FIGURA 136. PRUEBAS MODULARES. PRUEBA DE APLICACIÓN CLIENTE ANDROID: PRUEBA DE GESTIÓN DE AVISOS. CONSULTA DE HISTORIAL DE AVISOS REALIZADOS EL ÚLTIMO MES, INCLUYENDO FOTOS Y TICKETS FIRMADOS POR EL CLIENTE. 98
FIGURA 137. ANEXO I: MANUAL DE INSTALACIÓN. PAQUETE DE INSTALACIÓN DE LA APLICACIÓN CLIENTE DE GESTIÓN. IV
FIGURA 138. ANEXO I: MANUAL DE INSTALACIÓN. ACCESOS DIRECTOS DE LA APLICACIÓN CLIENTE DE GESTIÓN. IV
FIGURA 139. ANEXO I: MANUAL DE INSTALACIÓN. INSTALACIÓN DE LA APLICACIÓN ANDROID. V

Glosario

API.*Application Programming Interface*. Interfaz de programacion de aplicaciones, es el conjunto de funciones y procedimientos que ofrece una biblioteca para ser utilizado por otro software como una capa de abstracción.

Backup (o copia de seguridad). Copia de los datos originales que se realiza con el fin de disponer de un medio de recuperarlos en caso de pérdida.

DAO *Data Access Object***.** Componente de software que suministra una interfaz común entre la aplicación y uno o más dispositivos de almacenamiento de datos, tales como una base de datos o un archivo.

DTO *Data Transfer Object***.** Patrón de diseño utilizado para transferir datos entre subsistemas de aplicaciones de software. Los DTOs se usan habitualmente junto con los DAOs para recuperar datos de la base de datos.

ERP *Enterprise Resource Planning*. Sistema de planificación de recursos empresariales que integran y manejan muchas de las operaciones de producción y los aspectos de distribución de una compañía en la producción de bienes o servicios.

ETL *Extract,Transform and Load***.** Proceso que permite mover datos desde múltiples fuentes, reformatearlos y limpiarlos, y cargarlos en otra base de datos.

Framework. Estructura conceptual y tecnológica de soporte definido que puede servir de base para la organización y desarrollo de software. Típicamente, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto.

GPS. *Global Position System*. Es un sistema global de navegación por satélite que permite determinar la posición de un objeto en todo el mundo.

Hosting (o alojamiento Web). Servicio que provee a los usuarios de Internet un sistema para poder almacenar información, imágenes, vídeo, o cualquier contenido accesible vía Web.

JSON. *JavaScript Object Notation*. Formato ligero de intercambio de datos, completamente independiente del lenguaje de programación.

LINQ. *Language Integrated Query***.** Componente de la plataforma Microsoft .Net que agrega capacidades de consulta a datos de manera nativa a los lenguajes .Net.

MDI *Multiple Document Interface***.** Interfaz cuyas ventanas se encuentran dentro de una ventana padre (normalmente con la excepción de las ventanas modales).

ORM *Object-Relational Mapping***.** Técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional, utilizando un motor de persistencia.

Polling. Operación de consulta constante para crear una actividad sincrónica sin el uso de interrupciones.

Push email. Sistema de email en el que el nuevo email es transferido a su llegada al servidor de correo.

PYMES. *Pequeñas Y Medianas Empresas.* Empresas de tamaño pequeño o medio.

SAT. *Servicio de Atención Técnica.* Departamento o empresa encargada del mantenimiento o reparación de productos.

Smartphone (o teléfono inteligente). Teléfono móvil construido sobre una plataforma informática móvil, con una mayor capacidad de almacenar datos y realizar actividades semejantes a una mini computadora y conectividad que un teléfono móvil convencional.

SSH. *Secure Shell.* Nombre de un protocolo y del programa que lo implementa. Sirve para acceder a máquinas remotas a través de una red.

TIC. *Tecnologías de la Información y la Comunicación*. Conjunto de recursos, procedimientos y técnicas usadas en el procesamiento, almacenamiento y transmisión de información.

1 Introducción

1.1 Antecedentes

En junio de 2013, tras finalizar mis estudios de informática en la Universidad Autónoma de Madrid, presenté mi Trabajo de Fin de Grado "Alba Calvo Ruiz. Gestión de Trabajos en Movilidad (GTM). (COSITI_120)"[1], concebido con el fin de dar respuesta a las necesidades de gestión de empresas de mantenimiento y reparación en domicilios de clientes.

La versión inicial del proyecto GTM soportaba la siguiente funcionalidad:

- Gestión de clientes y avisos:
  - Gestión de clientes y avisos en la aplicación cliente corporativa .Net.
  - Envío de avisos a técnicos dotados de terminales en movilidad.
  - Recepción del trabajo realizado introducido por los técnicos en sus terminales de movilidad.

    Nota: el intercambio de información entre la aplicación corporativa y los terminales en movilidad corre a cargo de una aplicación Web PHP.

- Representación cartográfica de rutas de técnicos:
  - Tracking GPS de coordenadas obtenidas en terminales Android de los técnicos y anotadas en base de datos Web.
  - Presentación de mapas de rutas y ubicación de técnicos mediante aplicación Web PHP con utilización de Google Maps.

En el apartado "Trabajo futuro" de la memoria del proyecto se recogió una serie de posibles mejoras relativas a aspectos como:

- Posibilidad de trabajo offline desde terminales en movilidad.
- Mejoras en la representación cartográfica.
- Incorporación de funcionalidad de gestión documental (fotos, justificantes firmados por el cliente).

1.2 Objetivos

El objetivo del proyecto GTM v2.0 es la ampliación del proyecto GTM realizado en mi Trabajo de Fin de Grado, dando respuesta a algunas de las necesidades de mejora detectadas.

En concreto, se propone la ampliación del alcance de la aplicación, aportando mejoras como:

[1] http://ir.ii.uam.es/~fdiez/TFGs/gestion/leidos/2013/20130610AlbaCalvoRuiz.pdf

- Representación cartográfica avanzada de clientes, avisos y ubicación de técnicos integrada en la aplicación manejada por los operadores, útil de cara a la asignación de avisos a técnicos próximos, con el consecuente ahorro de tiempo.

- Representación cartográfica avanzada de rutas de técnicos integrada en la aplicación manejada por los operadores, permitiendo contrastar las paradas de los técnicos con los avisos asignados y comprobar la duración de las paradas mediante los justificantes de trabajos correspondientes firmados por el cliente.

- Posibilidad de trabajo en modo "desconectado" en los terminales en movilidad de los técnicos, sincronizando la base de datos y repositorio de ficheros locales del terminal en cuanto se disponga de conectividad.

- Subida de fotos relacionadas con el aviso desde los terminales en movilidad de los técnicos.

- Generación de justificantes de trabajo realizado firmados por los clientes en la pantalla de los terminales en movilidad de los técnicos.

- Gestión documental de ficheros asociados a clientes y avisos (justificantes, fotos).

1.3 Motivación

Tal y como comentaba en la memoria de mi Trabajo de Fin de Grado, tengo un gran interés en el área de la gestión empresarial y en el desarrollo de sistemas integrados por múltiples tecnologías.

Dado que en el proyecto GTM concurrían los dos aspectos anteriores, me pareció una buena elección en su momento.

La mejora de la versión inicial planteada en el proyecto actual me ha permitido profundizar en la solución de gestión empresarial inicial y realizar una implementación con una mayor integración de tecnologías como: aplicación corporativa .Net, aplicación cliente Android para terminales en movilidad, gestión de servidor de base de datos Web SQLServer, interoperabilidad entre base de datos local y base de datos Web mediante una API REST de servicios Web.

Otro poderoso motivo para la elección de este proyecto ha sido su posible utilización futura como embrión de un desarrollo de aplicación de negocio que creo podría contar con un nicho de mercado.

Existen numerosos productos estándar de movilidad integrables con diferentes ERP empresariales. También es posible para una empresa contratar una solución a medida integrable con su ERP. Estas soluciones pueden ser convenientes o incluso óptimas para empresas de tamaño medio o alto, pero creo que por cuestiones como el coste o la necesidad de disponer de un ERP corporativo quedan fuera del alcance de muchas PYMES de tamaño reducido.

Creo que para estas empresas podría resultar muy interesante una aplicación que incluyera:

- Gestión comercial: facturación, cobros, pagos, etc.
- Contabilidad soportada por la propia aplicación o mediante interoperabilidad con productos de contabilidad estándar.
- Gestión vertical de la problemática específica de la empresa (por ejemplo gestión de avisos para empresas de servicios).
- Gestión documental.
- Movilidad.
- Geolocalización y representación cartográfica.
- …

La disponibilidad de este tipo de productos es todavía escasa, ya que su desarrollo implica la respuesta a una problemática "vertical" específica de cada sector.

Me gustaría que el aprendizaje obtenido me abriera las puertas a un trabajo como desarrolladora de soluciones de gestión empresarial de características similares a la propuesta objeto de mi proyecto, con un alto grado de implicación y responsabilidad desde el primer momento.

1.4 Organización de la memoria

La memoria consta de seis capítulos y un anexo que se detallan a continuación:

- **Capítulo 1: Introducción**

  Explicación de antecedentes, objetivos y motivación del proyecto. También se detalla la estructura de la memoria.

- **Capítulo 2: Estado del arte**

  Descripción del estado actual de las tecnologías implicadas en el proyecto, incluyendo una comparativa de sus principales ventajas y desventajas.

- **Capítulo 3: Diseño**

  Resumen de la arquitectura del proyecto y breve explicación de cada una de sus partes, incluyendo detalles de su interfaz.

- **Capítulo 4: Desarrollo**

  Análisis de las tecnologías implicadas en la realización del proyecto y breve explicación del funcionamiento del mismo. Se incorporan además diagramas (UML, secuencia) para ilustrar mejor la explicación.

- **Capítulo 5: Integración, pruebas y resultados**

  Detalle de las pruebas realizadas (modulares, de integración, etc.).

- **Capítulo 6: Conclusiones y trabajo futuro**

  Exposición resumida del planteamiento y logro de los objetivos del proyecto y de su posible ampliación.

- **Anexo A: Manual de instalación**

  Explicación del procedimiento de instalación.

2 Estado del arte

2.1 Introducción

En los apartados siguientes se analiza la tecnología disponible en relación con los componentes del proyecto detallados en el capítulo 3 “Diseño".

En concreto se contempla la situación de las tecnologías relacionadas con:

- Desarrollo de aplicaciones cliente de escritorio.
- Aplicaciones cliente para terminales en movilidad.
- Servicios Web.
- Representación cartográfica.
- Servidores de bases de datos.

2.2 Aplicación cliente de escritorio

Una aplicación de escritorio puede ser desarrollada con múltiples tecnologías: C, C++, Delphi, Java, .Net (C# u otros lenguajes), Python, etc.

Las dos tecnologías a la cabeza actualmente son Java y .Net. [1]

A continuación se describen ambas opciones:

2.2.1 Java

2.2.1.1 Ventajas

- Dispone de varias APIs de desarrollo gráfico: AWT (Abstract Window Toolkit) con librerías Swing, SWT (Standard Widget Toolkit).[2]

- Alta portabilidad de sus aplicaciones.

- Es una plataforma libre.

2.2.1.2 Desventajas

- Tiene una mayor orientación a desarrollos WEB frente a .Net más orientado a aplicaciones cliente, presentando una menor productividad que .Net en el desarrollo de interfaces de usuario en el ámbito de aplicaciones cliente para PCs.

- El diseño de interfaces gráficas con AWT y Swing no resulta trivial.

[2] http://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n)#En_aplicaciones_de_escritorio

2.2.2 .Net

2.2.2.1 Ventajas

- Dispone de varias tecnologías de creación de interfaces: Windows Forms, Windows Presentation Foundation.[3]

- Tiene una fuerte orientación al desarrollo de aplicaciones cliente para PC, presentando una alta productividad en este entorno.

2.2.2.2 Desventajas

- Actualmente presenta poca portabilidad de aplicaciones, aunque se están llevando a cabo proyectos como "Mono"[4] para hacer .Net compatible con otros sistemas operativos.

- Es una plataforma de desarrollo propietaria que debe ser licenciada bajo pago.

2.3 Plataformas de referencia para terminales en movilidad.

Las cuatro plataformas de mayor implantación son: [2]

- Android de Google
- iOS de Apple
- Blackberry OS de Blackberry (antes RIM)
- Windows Phone de Microsoft

A continuación se analiza en detalle cada una de ellas:

2.3.1 Windows phone

2.3.1.1 Ventajas

- Utiliza una potente herramienta de desarrollo basada en .Net Compact Framework.[5]

2.3.1.2 Desventajas

- Es una incógnita si alcanzará un grado de operatividad, eficiencia en el aprovechamiento del hardware e implantación suficiente.

2.3.2 Blackberry OS

2.3.2.1 Ventajas

- Proporciona servicios avanzados de comunicaciones basados en plataforma propietaria.

[3] http://msdn.microsoft.com/es-es/library/w0x726c2(v=vs.100).aspx
[4] http://www.mono-project.com/Main_Page
[5] https://es.wikipedia.org/wiki/Windows_Phone#Desarrollo_de_aplicaciones

Especialmente interesante resulta el servicio de "Push email"[6], que proporciona inmediatez en el correo entrante sin necesidad de realización de "polling" (consulta constante) desde el terminal. [3]

2.3.2.2 Desventajas

- "Push email", a pesar de ser un gran servicio, va perdiendo peso frente al conjunto de servicios avanzados ofrecidos por otros tipos de terminal. Por otra parte, la recepción inmediata de emails mediante "polling" utilizando otros terminales ofrece un funcionamiento correcto sin incremento de coste (el consumo de ancho de banda no es significativo).

- El futuro de la empresa está sujeto a cambios e incertidumbres.

  Recientemente se ha producido un cambio de plataforma (Blackberry 10) y de herramienta de desarrollo (Java RIM hacia C++). [7]

2.3.3 iOS

2.3.3.1 Ventajas

- Es líder junto a Android en prestaciones y disponibilidad de aplicaciones, así como en implantación y número de usuarios.[8]

2.3.3.2 Desventajas

- Es una solución cerrada con terminales de Apple. Esto puede ser un problema para servicios en movilidad que requieran tipos específicos de terminales (de bajo coste, con equipamiento específico como lectores de código de barras hardware, impresoras…).

- Es necesario certificar todas las aplicaciones en cada una de sus versiones e incluirlas en el AppStore de Apple. Esto puede ser un inconveniente para pequeños desarrollos.

- Utiliza un lenguaje de desarrollo específico y poco extendido (ObjectiveC) con la correspondiente curva de aprendizaje.[9]

2.3.4 Android

2.3.4.1 Ventajas

- Es líder junto a iPhone en prestaciones y disponibilidad de aplicaciones así como en implantación y número de usuarios (92% de los nuevos smartphones en España funcionan con Android[10]).

---

[6] http://es.wikipedia.org/wiki/Tecnolog%C3%ADa_Push
[7] http://alt1040.com/2013/02/thorsten-heins-y-la-era-blackberry
[8] http://www.24horas.cl/tendencias/mundodigital/iphone-lidera-en-eeuu-y-android-es-el-lider-mundial-481300
[9] http://es.wikipedia.org/wiki/Apple_iOS#Contenido_del_SDK
[10] http://cincodias.com/cincodias/2013/04/17/tecnologia/1366205805_952527.html

- Su sistema operativo es de libre distribución y se encuentra implantado en una gama muy amplia de terminales. Esto puede ser una ventaja significativa en servicios profesionales que requieran terminales de bajo coste, con hardware específico, etc.

- Utiliza un lenguaje de desarrollo Java con una curva de aprendizaje rápida y entornos de desarrollo de libre distribución.

2.3.4.2 Desventajas

- Debido a la proliferación de fabricantes que lo implementan aparece el problema de "fragmentación de terminales" debido a los cambios introducidos en el sistema operativo por algunos de estos fabricantes. Esto provoca que los programas desarrollados para un modelo de terminal no sean totalmente compatibles con el resto.[11]

2.4 Servicios Web

Existen dos tecnologías a la cabeza en el desarrollo de servicios Web [4] [5]:

- SOAP (Simple Object Access Protocol)[12]. Es un protocolo estándar de comunicación entre diferentes procesos mediante intercambio de datos XML.

- REST (Representational State Transfer)[13]. Es un estilo de arquitectura software para sistemas distribuidos centrado en los estándares HTTP y XML o JSON para la transmisión de datos.

A continuación se analiza en detalle cada una de ellas:

2.4.1 SOAP

2.4.1.1 Ventajas

- Independiente de lenguaje, plataforma y protocolo de transporte.

- Es el estándar para la realización de servicios Web y dispone de una gran variedad de herramientas y un amplio soporte.

2.4.1.2 Desventajas.

- Conceptualmente más difícil y más pesado que REST.

- Más difícil de desarrollar, dado que requiere el uso de herramientas.

- Utiliza XML como contenedor de información, formato que puede dar problemas cuando se dispone de pocos recursos, como es el caso de las aplicaciones móviles.

[11] http://www.gacetatecnologica.com/movilizate/?p=236
[12] http://es.wikipedia.org/wiki/Simple_Object_Access_Protocol
[13] http://es.wikipedia.org/wiki/Representational_State_Transfer

2.4.2 REST

2.4.2.1 Ventajas

- Independiente de lenguaje y plataforma.
- Desarrollo más sencillo que con SOAP.
- Curva de aprendizaje baja.
- Menor dependencia de herramientas.
- Permite utilizar JSON en vez de XML como contenedor de la información, muy útil cuando se dispone de recursos escasos en el caso de dispositivos móviles.

2.4.2.2 Desventajas.

- Utiliza un modelo de comunicación punto a punto, no aplicable a un entorno distribuido donde el mensaje pueda ir a través de uno o más intermediarios.
- Ligado al protocolo de transporte HTTP.

2.5 Representación cartográfica

Para la presentación de mapas desde la aplicación cliente de escritorio se han contemplado dos opciones:

- Google Maps[14]. Es un servidor de aplicaciones de mapas en la Web que pertenece a Google. Ofrece imágenes de mapas desplazables, así como fotografías por satélite del mundo y rutas entre diferentes ubicaciones.
- Microsoft MapPoint[15]. Es una aplicación cliente de representación cartográfica que permite pintar información geoespacial del mundo, geolocalizar direcciones y dibujar rutas.

A continuación se analiza en detalle cada una de las opciones:

2.5.1 Google Maps

2.5.1.1 Ventajas

- Dispone de una API ampliamente documentada.
- Ofrece tres tipos de vistas de mapas: vista estándar de calles y carreteras, imagen satélite y vista de terreno.

[14] https://maps.google.es/
[15] http://www.microsoftstore.com/store/mseea/es_ES/pdp/MapPoint-2013-Europe/productID.278339000

2.5.1.2 Desventajas

- Alto coste. Si se cobra el acceso a la solución que utiliza Google Maps, es necesario adquirir la licencia para la API "Google Maps for Work", que tiene un coste de unos 8000€ al año.[16]

- La licencia de la API "Google Maps for Work" presenta un límite de 100.000 solicitudes diarias.

- La licencia de Google Maps no permite el almacenamiento de los datos de geolocalización de forma permanente. Sólo permite guardar la información en caché, pero no almacenarla en una base de datos interna y reutilizarla con otros propósitos diferentes.

2.5.2 MapPoint

2.5.2.1 Ventajas

- Mayor velocidad de operación ante una base de datos con un gran número de direcciones que geolocalizar en el mapa al presentar una menor latencia que los servicios Web de Google Maps.

- Instalación local en el equipo. No precisa de conexión a Internet como los servicios Web de Google.

- No presenta un límite de solicitudes diarias de geolocalización y es capaz de resolver mayor cantidad de solicitudes que Google Maps.

- Fácil integración con .Net mediante la librería de MapPoint, disponible una vez instalado el producto.

- Bajo coste. La licencia de MapPoint cuesta 231€ para cada equipo en el que se instala.

- Ofrece varios tipos de vistas de mapas: mapa de carreteras, de terreno, estadísticos, etc.

2.5.2.2 Desventajas

- Necesidad de instalación de MapPoint en cada equipo que utilice la representación cartográfica.

- Necesidad de adquirir una licencia para cada equipo usuario.

[16] http://www.geocoderpro.com/en/resources/google-maps-licensing/

2.6 Base de Datos.

Existen múltiples servidores de bases de datos en el mercado actual.

Por coste y tamaño quedan fuera del ámbito de este proyecto productos como DB2, Oracle, etc.

A continuación se evalúan varios servidores de gran implantación, que presentan un coste asumible (en algunos casos nulo por tratarse de productos de carácter gratuito) y una capacidad de almacenamiento acorde a las necesidades de GTM: SQLServer, MySql, y PostgreSql. [6]

2.6.1 SQLServer

2.6.1.1 Ventajas

- Se trata de un producto consolidado acompañado de una serie de herramientas de mantenimiento y productividad que aportan un gran valor añadido como:
  - Cubos OLAP, etc a la altura de grandes bases de datos.[17]
  - Management Studio con soporte de tareas de mantenimiento de la base de datos, de backup, etc.
  - Agente de programación de tareas (de backup, etc.).
  - Sistema de integración de datos externos (SSIS).
  - Transact SQL.[18]

- Integración con herramientas Microsoft (.Net, etc).

- Facilidad de aprendizaje.

2.6.1.2 Desventajas

- Es un producto de pago que es necesario licenciar.

2.6.2 MySQL.

2.6.2.1 Ventajas

- Es un producto consolidado de gran rapidez.

- Libre distribución.

2.6.2.2 Desventajas

- Su adquisición por parte de Oracle condiciona su futuro a las posibles decisiones de esta corporación.[19] Esto está provocando el desplazamiento de usuarios a otras alternativas como MariaDB. [7]

[17] http://es.wikipedia.org/wiki/Cubo_OLAP
[18] http://es.wikipedia.org/wiki/Transact-SQL
[19] http://www.espaciolinux.com/2010/01/aprobada-compra-sun-oracle-mysql-java-parte-trato/

- No cuenta con herramientas de mantenimiento similares en productividad a Management Studio de SQLServer. Tampoco cuentan con herramientas ETL (Extract, Transform, Load)[20] para la extracción de datos de fuentes externas, reformateo y carga en la base de datos, mientras que SQLServer cuenta con SSIS (SQLServerIntegrationServices).[21]

2.6.3 PostgreSQL

2.6.3.1 Ventajas

- Se trata de un producto consolidado y es una alternativa real a MySQL y SQLServer.

- Libre distribución.

2.6.3.2 Desventajas

- Herramientas menos amigables que las de SQLServer y MySQL.

- .Net posee una fuerte integración con el ORM Entity Framework para SQLServer con mucha documentación y ejemplos disponibles. No he encontrado referencias a ORMs para PostgreSQL integrables con .Net con un grado de difusión y documentación equiparable al de Entity Framework.

[20] http://es.wikipedia.org/wiki/Extract,_transform_and_load
[21] http://msdn.microsoft.com/en-us/library/ms141026.aspx

3 Diseño

3.1 Arquitectura

El proyecto GTM a la finalización de mi Trabajo de Fin de Grado contaba con una arquitectura con dos partes fundamentales:

- Subsistema de gestión de clientes y avisos incluyendo alta y asignación de avisos a técnicos desde la aplicación corporativa.Net y cumplimentación de avisos por parte de los técnicos desde sus terminales en movilidad:

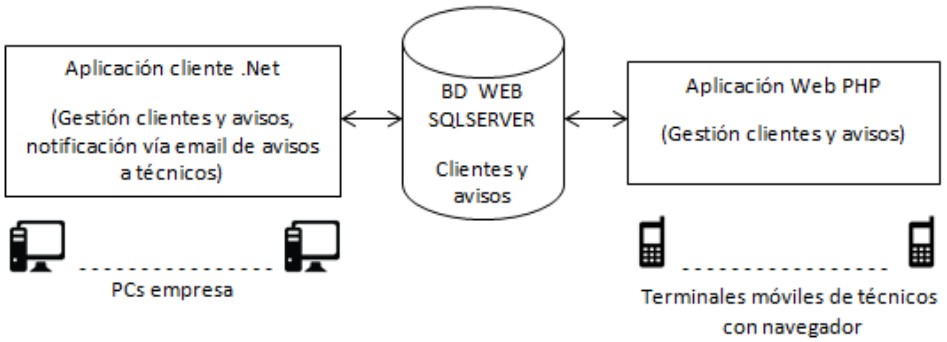


Figura 1. Esquema GTM v1.0. Subsistema de gestión de clientes y avisos.

- Subsistema de envío y almacenamiento de coordenadas de tracking GPS de los técnicos desde sus terminales Android y de representación de las rutas seguidas por los mismos desde la aplicación Web utilizando Google Maps:

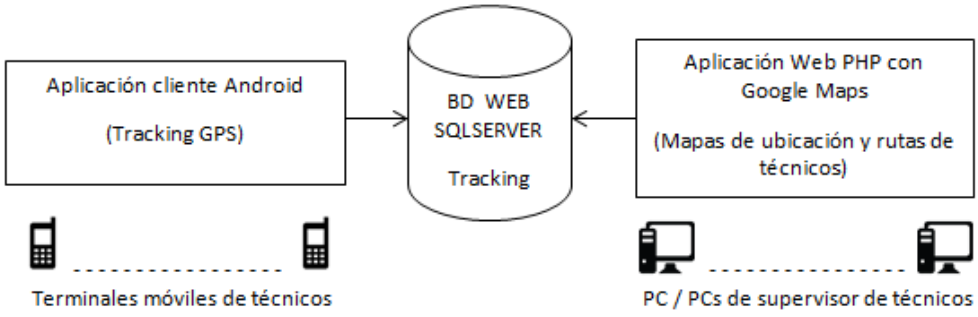


Figura 2. Esquema GTM v1.0. Subsistema de tracking GPS y representación cartográfica.

A partir de las mejoras detectadas y detalladas en el apartado 1.2 del documento, se propone la siguiente arquitectura para la versión 2.0 del proyecto GTM:

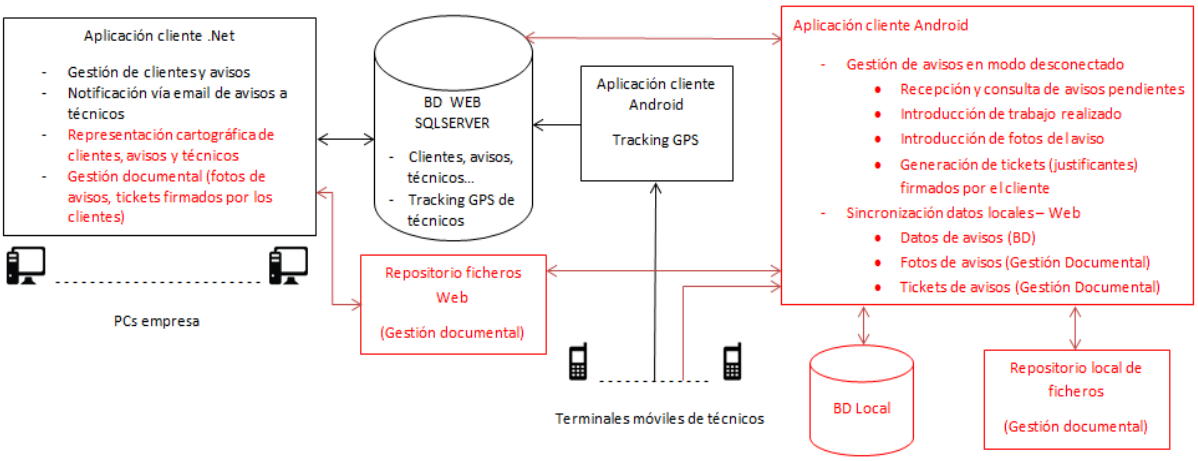


Figura 3. Esquema de nueva funcionalidad GTM v2.0.

La nueva funcionalidad soportada por la aplicación es la siguiente:

➢ <u>Aplicación cliente .Net:</u>

- Representación cartográfica
  - Clientes
    - Geolocalización (obtención de coordenadas geográficas) y representación de mapa de ubicación de las direcciones de clientes utilizando MapPoint.
    - Apertura de ficha de cliente mediante click en su marcador.
  - Avisos
    - Mapa de avisos pendientes asignados a un técnico y avisos pendientes sin asignar.
    - Mapa de ruta seguida por un técnico en una fecha junto con los avisos realizados por el técnico en esa fecha.

      <u>Nota:</u> Utilización de datos procedentes del tracking GPS (rutas de técnicos) y de las coordenadas de los clientes geolocalizados a los que se realizan los avisos (marcadores de avisos).
    - Apertura de ficha de aviso mediante click en el marcador del aviso.

- Gestión documental
  - Subrejilla de documentos de un cliente.
  - Rejilla de documentos de un aviso.
  - Rejilla general de documentos.
  - Posibilidad de apertura del documento seleccionado desde cualquiera de las rejillas.
  - Posibilidad de filtro por tipo de documento (foto/ticket) en todas las rejillas (funcionalidad estándar de todas las rejillas de la aplicación).

- Aplicación cliente Android:

  - Gestión de avisos en modo desconectado
    - Consulta de avisos pendientes con advertencia de avisos nuevos.
    - Introducción de trabajo realizado mediante edición del aviso.
    - Introducción de fotos de avisos.
      - Lanzamiento de toma de foto desde la ficha del aviso.
      - Proceso de toma de foto embebido en la aplicación con almacenamiento automático en gestión documental local (como documento tipo foto hijo del aviso) al final del proceso.
      - Consulta de fotos local desde la ficha del aviso con posibilidad de eliminación de fotos a descartar.
    - Generación de tickets (justificantes) de trabajo realizado firmados por el cliente.
      - Presentación de un resumen del aviso (fecha, trabajo realizado, etc.) en una pantalla con posibilidad de firma por parte del cliente.
      - Grabación del ticket en gestión documental local (como documento tipo ticket hijo del aviso).

  - Sincronización de datos locales y Web
    - Datos de avisos
      - BD Web a BD local: nuevos avisos asignados al técnico.
      - BD local a BD Web: datos de cumplimentación de avisos (fecha realización, trabajo realizado, etc.).
    - Ficheros (fotos y tickets)
      - Repositorio local a repositorio Web: copia de ficheros desde repositorio local a Web.
      - BD local a Web: registro de localización de ficheros "subidos" (rutas relativas) en BD Web.

De esta nueva arquitectura se deriva la eliminación de la aplicación Web, transportando la representación cartográfica a la aplicación cliente .Net y la gestión de avisos por parte de los técnicos, que ahora además permite trabajo offline, a la aplicación móvil.

Se incluye además una API de servicios Web para la interoperabilidad de la aplicación Android con el servidor de base de datos Web.

3.2 Esquema de base de datos

GTM cuenta con una base de datos alojada en un servidor SQL Server que permite el alta, la modificación y la eliminación de usuarios, empleados, clientes y avisos desde la aplicación cliente de gestión.

También posibilita la gestión de documentos (justificantes, fotos) asociados a los avisos y el almacenamiento de coordenadas de localización desde la aplicación móvil de tracking GPS.

La base de datos cuenta con la siguiente estructura de tablas:

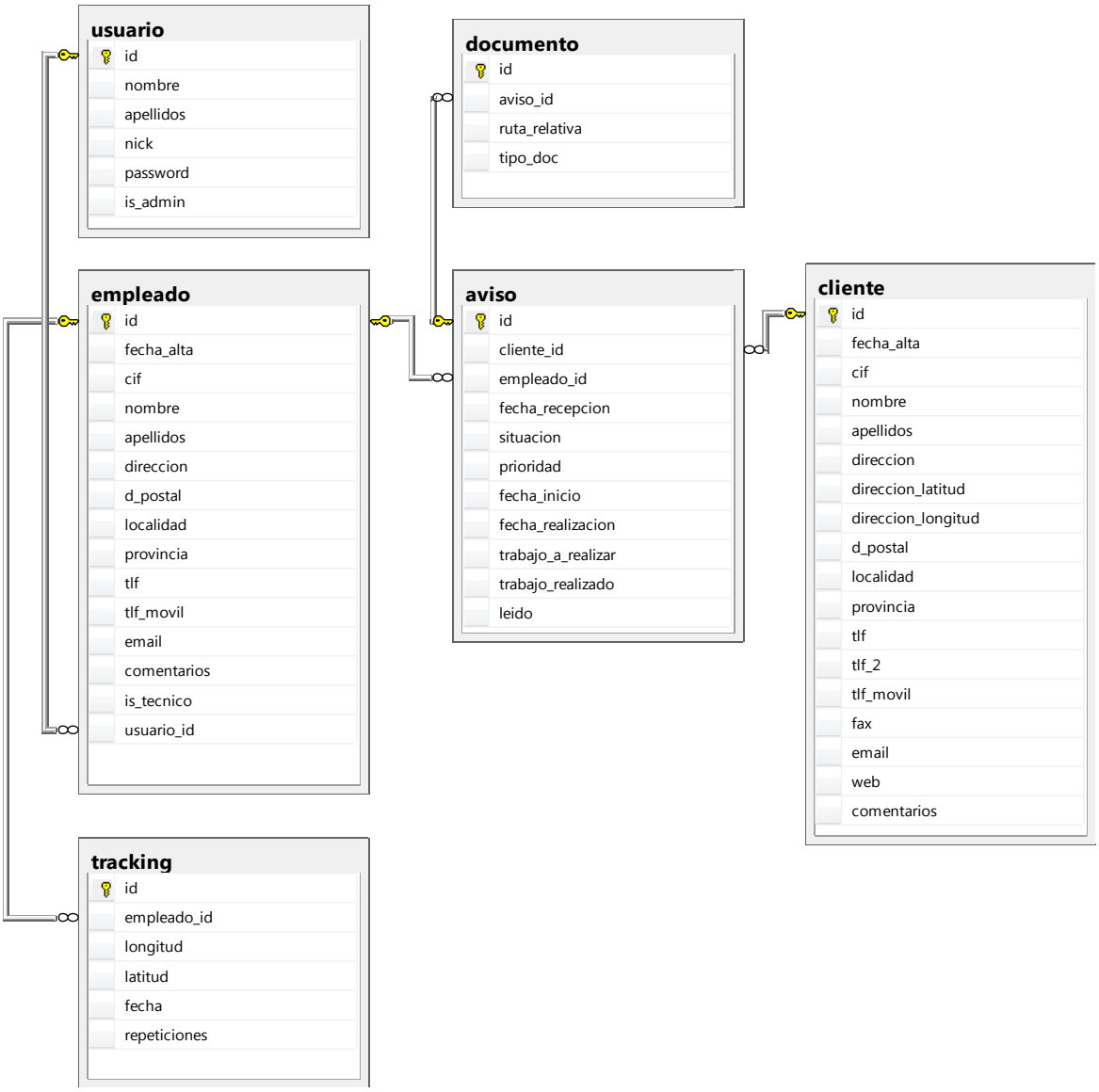


Figura 4. Esquema de la base de datos.

Se puede observar que hay tres roles de usuario: empleado operador, empleado técnico (empleado con el flag is_tecnico = true) y administrador (usuario con el flag is_admin = true). Estos roles permiten la identificación selectiva de usuarios dependiendo de la aplicación a la que acceden. Así, la aplicación corporativa de gestión y representación cartográfica sólo puede ser utilizada por usuarios con rol de empleado operador o administrador, diferenciando contenido según el rol, y la aplicación móvil de tracking y gestión de avisos sólo puede ser utilizada por usuarios con rol de técnico.

El esquema también incluye la tabla aviso, asociada con un empleado técnico y un cliente determinados. Además, el aviso puede tener documentos asociados en caso de que el técnico suba fotos de la realización del aviso o genere un justificante de trabajo.

Por último, también existe una tabla tracking que almacena las coordenadas de geolocalización de un técnico enviadas desde su terminal móvil.

3.3 Interfaz de usuario

3.3.1 Aplicación cliente de gestión

El objetivo de la aplicación es la gestión de avisos y su asignación a técnicos. Además, contempla la gestión de clientes y usuarios de la aplicación.

La aplicación cuenta con dos roles de usuario:

- El administrador, encargado de gestionar (dar de alta, modificar o eliminar) clientes, empleados (operadores y técnicos) y usuarios. Además, puede gestionar avisos asignando rutas a los técnicos basándose en la localización de los avisos y comprobar el recorrido de los mismos comparándolo con la realización de avisos asignados.

- El empleado, con un tipo de acceso más restringido y con la única posibilidad de gestionar avisos como se explica en el punto anterior.

En esta aplicación se darán de alta los avisos, que se notificarán al técnico mediante el envío de un correo electrónico a su cuenta de la empresa. Estos avisos podrán ser visualizados y gestionados posteriormente por los técnicos mediante la aplicación móvil.

A continuación se muestran las pantallas de la aplicación:

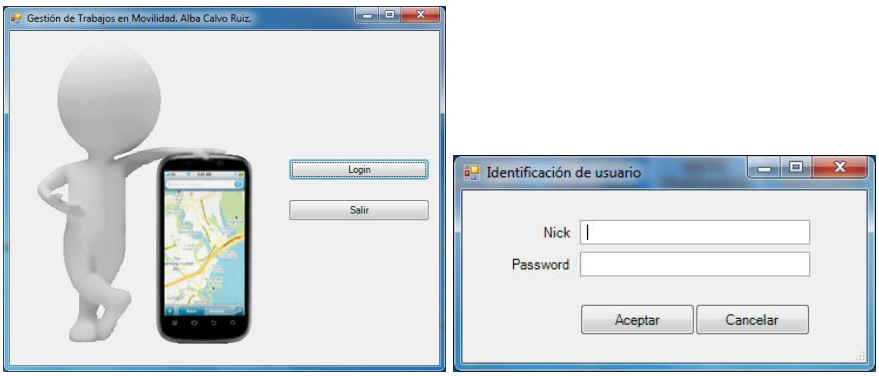


Figura 5. Aplicación cliente .Net: Pantalla de login.

Inicialmente se muestra la pantalla de login. Si los datos no son correctos se muestra un mensaje de error y si lo son el flujo del programa puede seguir dos caminos según el tipo de usuario autentificado:

3.3.1.1 Empleado

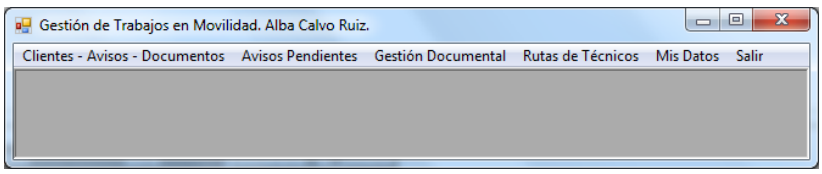


Figura 6. Aplicación cliente .Net: Menú de empleado.

El empleado dispone de varias opciones:

- “Clientes – Avisos - Documentos”: gestión de clientes y avisos, y de sus documentos asociados.
- “Avisos Pendientes”: gestión y presentación de mapas de avisos pendientes asignados a un técnico y de avisos pendientes sin asignar.
- “Gestión Documental”: consulta de documentos (fotos, tickets) asociados a avisos y clientes con posibilidad de apertura y visualización de los mismos.
- “Rutas de técnicos”: presentación de recorridos de técnicos y de avisos realizados para una fecha dada.
- “Mis Datos”: cambio de contraseña.
- “Logout”: cierre de sesión.

Esto se resume en el siguiente diagrama de estados:

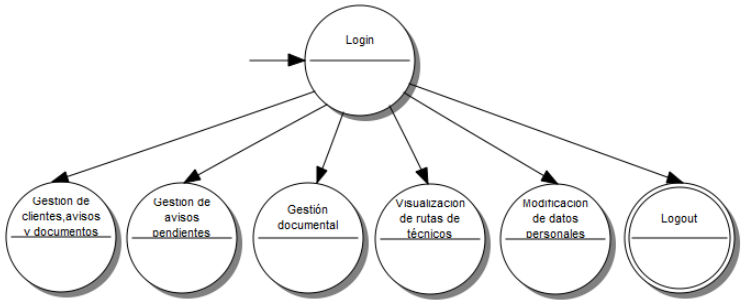


Figura 7. Aplicación cliente .Net: Máquina de estados de empleado.

Toda esta funcionalidad está comprendida dentro de la del usuario con rol de administrador, por lo que se describirá en detalle más adelante.

3.3.1.2 Administrador

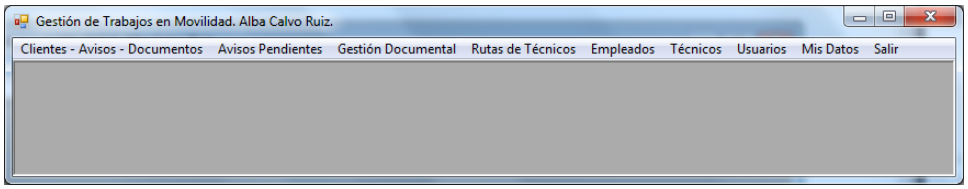


Figura 8. Aplicación cliente .Net: Menú de administrador.

El administrador dispone de varias opciones:

- “Clientes – Avisos - Documentos”: gestión de clientes y avisos, y de sus documentos asociados.
- “Avisos Pendientes”: gestión y presentación de mapas de avisos pendientes asignados a un técnico y de avisos pendientes sin asignar.
- “Gestión Documental”: consulta de documentos (fotos, tickets) asociados a avisos y clientes con posibilidad de apertura y visualización de los mismos.
- “Rutas de técnicos”: presentación de recorridos de técnicos y de avisos realizados para una fecha dada.
- “Empleados”: gestión de empleados operadores de la empresa.
- “Técnicos”: gestión de técnicos de la empresa.

- “Usuarios”: gestión de usuarios de la solución GTM.
- “Mis Datos”: cambio de contraseña y datos personales.
- “Logout”: cierre de sesión.

Al seleccionar cualquiera de las pestañas anteriores se abre su formulario asociado.

Esto se resume en el siguiente diagrama de estados:

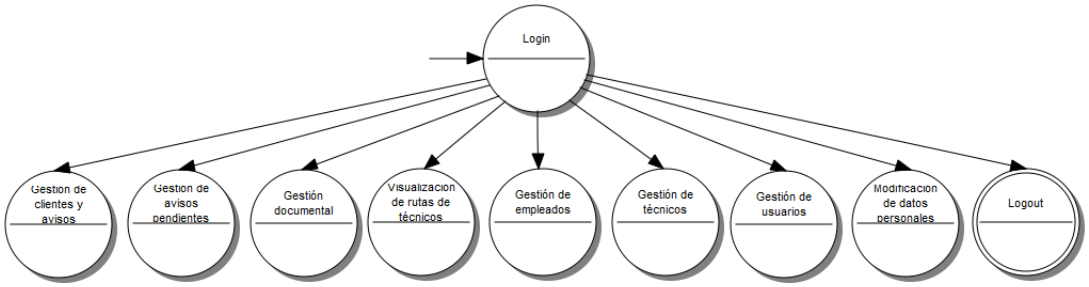


Figura 9. Aplicación cliente .Net: Máquina de estados de administrador.

A continuación se expone un resumen del comportamiento de cada uno de los formularios asociados:

3.3.1.2.1 Clientes – Avisos – Documentos

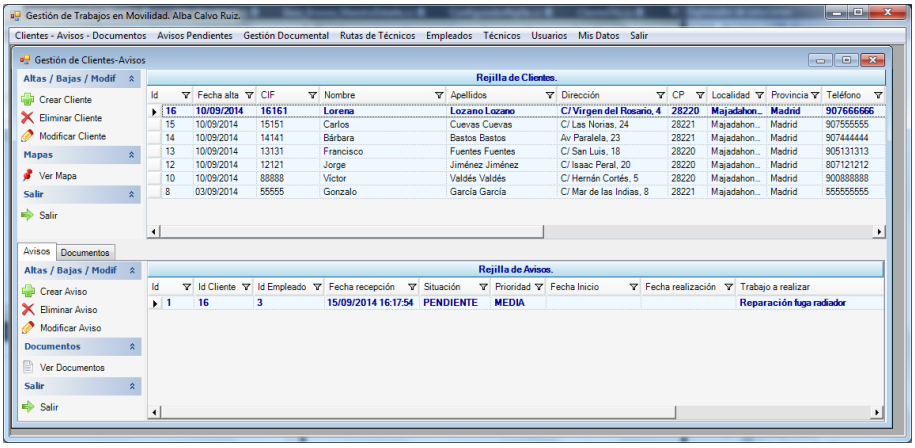


Figura 10. Aplicación cliente .Net: Ventana de Clientes-Avisos-Documentos.

Este formulario es una ventana maestro/detalle en la que se muestran los datos de clientes en la rejilla superior (maestro) y los datos de avisos y documentos asociados a cliente en las pestañas de la rejilla inferior (detalle).

Los datos del detalle dependen del ítem seleccionado en la rejilla maestro.

Las rejillas incorporan una barra lateral izquierda que muestra las acciones disponibles: alta, eliminación y modificación de registros en todas ellas, así como presentación de mapa de clientes en el maestro, visualización de documentos asociados a aviso en el detalle de avisos y visualización de documentos asociados a cliente en el detalle de documentos.

Además, las rejillas también incluyen la funcionalidad de filtrado de columnas estándar presente en todos los formularios de la aplicación.

- ***Altas, bajas y modificaciones.***

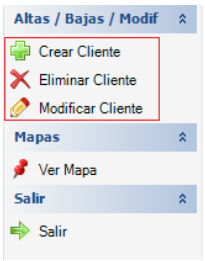


Figura 11. Aplicación cliente .Net: Altas, bajas, modificaciones.

El funcionamiento de las altas, bajas y modificaciones es idéntico en todos los formularios. La elección de cualquiera de las tres opciones supone la apertura de un nuevo formulario con campos editables que presenta un aspecto distinto según la opción escogida.

A continuación se ilustra el ejemplo de un alta de un aviso para el cliente seleccionado (16, Lorena Lozano), asignándole un técnico y estableciendo su prioridad:

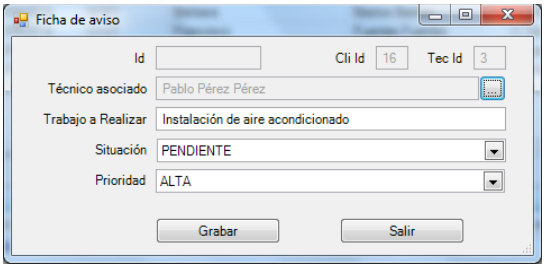


Figura 12. Aplicación cliente .Net: Alta de aviso.

La ficha consta de una serie de campos a rellenar. Al finalizar es necesario elegir la opción “Grabar” si se desea guardar los cambios de forma permanente.

Al grabar el formulario, el alta se ve reflejada en la rejilla de avisos, donde también se puede ver el historial completo de avisos de ese cliente:

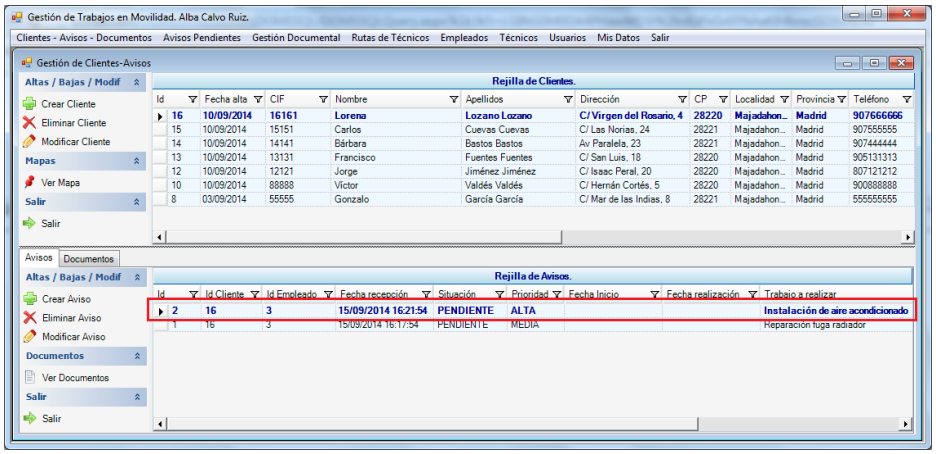


Figura 13. Aplicación cliente .Net: Rejilla de avisos con cambios reflejados.

Cabe destacar que cada vez que cada vez que se realice el alta de un aviso asociado a un técnico, éste recibirá una notificación en su correo de la empresa:

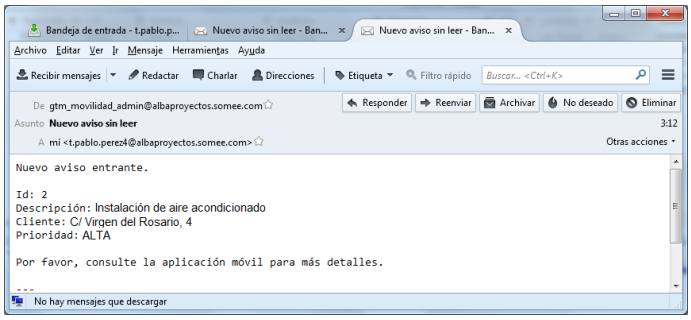


Figura 14. Aplicación cliente .Net: Mensaje de notificación de recepción de aviso.

La funcionalidad de modificación y eliminación es muy similar, inhabilitando la edición de campos en el caso de la eliminación, por lo que no se incluirán imágenes explicativas de las mismas.

- ***Mapa de clientes.***

Figura 15. Aplicación cliente .Net: Ver mapa de clientes.

Al seleccionar la opción de visualización de mapas de clientes, se presenta al usuario un mapa con la situación de todos los clientes de la aplicación. Para ello se utilizan las librerías para .Net de MapPoint.

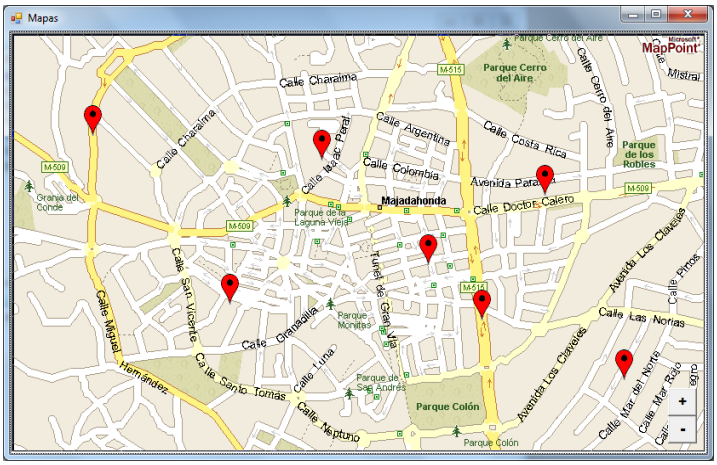


Figura 16. Aplicación cliente .Net: Mapa de clientes de la aplicación.

Al seleccionar un marcador, se muestra un formulario emergente con la información del cliente:

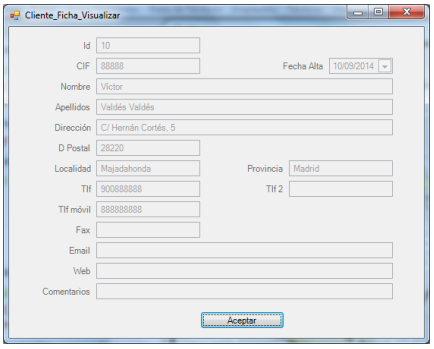


Figura 17. Aplicación cliente .Net: Información de cliente.

- ***Visualización de documentos asociados a un aviso.***

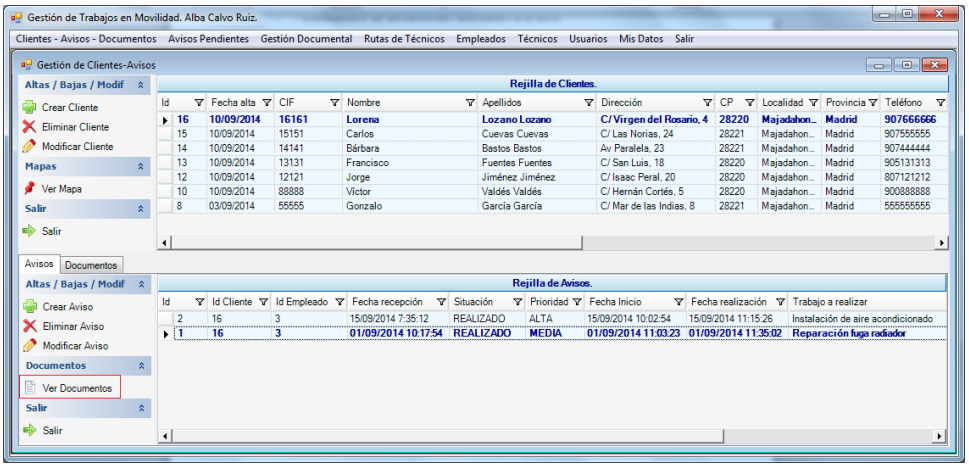


Figura 18. Aplicación cliente .Net: Documentos asociados a un aviso.

Es posible visualizar los documentos asociados a un aviso seleccionando esta opción. Esto presenta al usuario un formulario con la información de los

documentos asociados, incluyendo la ruta del servidor Web donde se aloja el documento y el tipo del mismo (FOTO o TICKET):

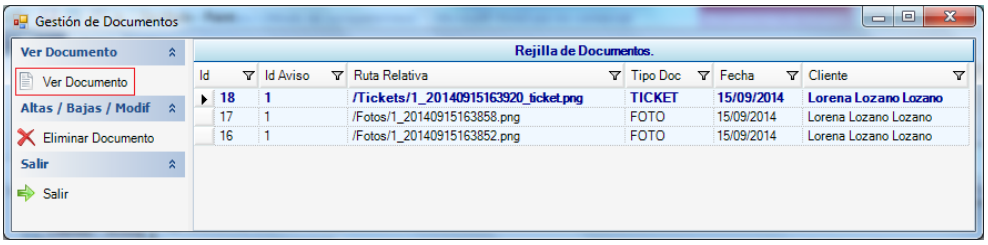


Figura 19. Aplicación cliente .Net: Visualización de un documento.

El usuario dispone entonces de dos opciones:

- Visualización del documento: presenta una vista previa del documento.

Figura 20. Aplicación cliente .Net: Documento (foto y ticket, respectivamente).

- Eliminación del documento: la selección de esta opción muestra al usuario un mensaje de confirmación y elimina el documento completamente del servidor Web.

➢ *Visualización de documentos asociados a un cliente.*

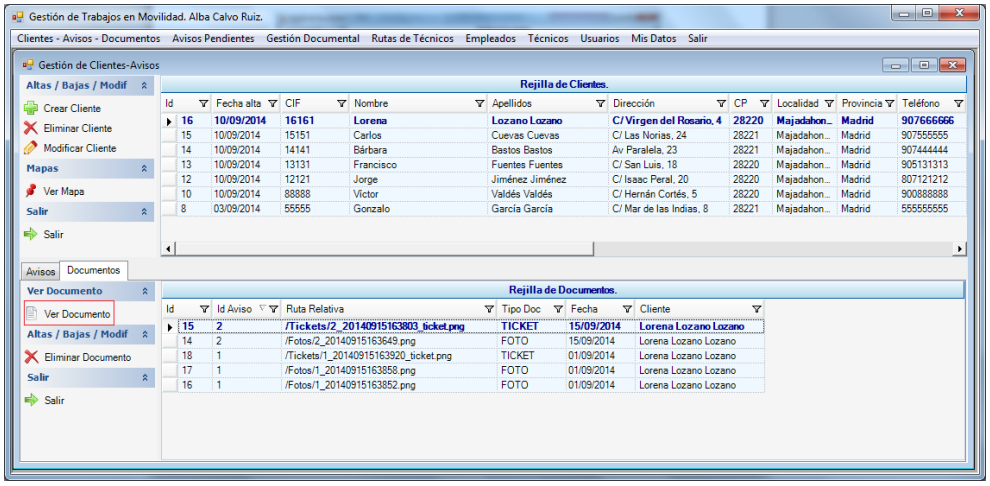


Figura 21. Aplicación cliente .Net: Documentos asociados a un cliente.

El comportamiento de la rejilla es idéntico al de la opción anterior. En este caso, se presentan los avisos asociados al cliente seleccionado en el maestro en lugar de los asociados a un aviso.

3.3.1.2.2 Avisos pendientes

En esta opción se muestran todos los avisos pendientes de realización asignados a un técnico y los avisos pendientes sin asignar.

Aunque al recibir y dar de alta un aviso en la aplicación es posible asignarle un técnico, como se ilustra en el punto anterior, también es posible dejar el aviso sin asignar y utilizar el presente formulario para incluirlo en el recorrido del técnico con avisos ya asignados próximos.

Así, al dar de alta un aviso desde la rejilla de "Clientes-Avisos-Documentos" el técnico se dejaría vacío:

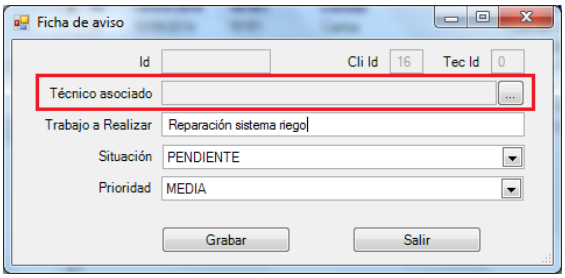


Figura 22. Aplicación cliente .Net: Alta de aviso sin técnico asociado.

Los avisos sin técnico asignado se indican en rojo en el detalle de avisos de la rejilla "Clientes-Avisos-Documentos":

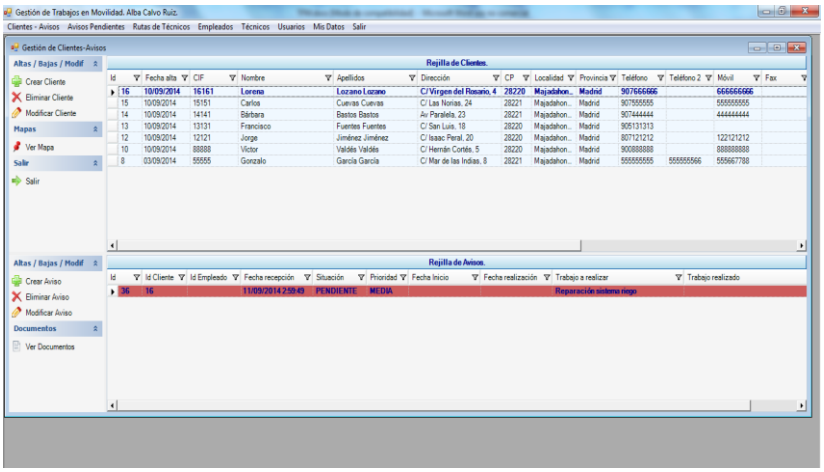


Figura 23. Aplicación cliente .Net: Presentación de avisos sin técnico asociado en rojo en la rejilla de detalle del formulario Clientes-Avisos-Documentos.

A continuación, se accede al menú "Avisos Pendientes". En esta rejilla se pueden ver tanto los avisos asignados a técnicos como los pendientes de asignar en rojo.

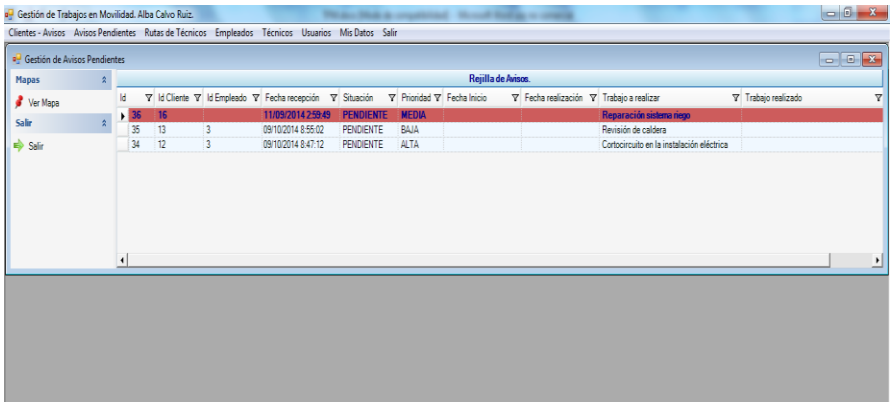


Figura 24. Aplicación cliente .Net: Rejilla de avisos pendientes.

Si se selecciona la opción "Ver Mapa", se pueden ver en gris los avisos sin técnico asignado y en azul los que ya están asignados a técnico:

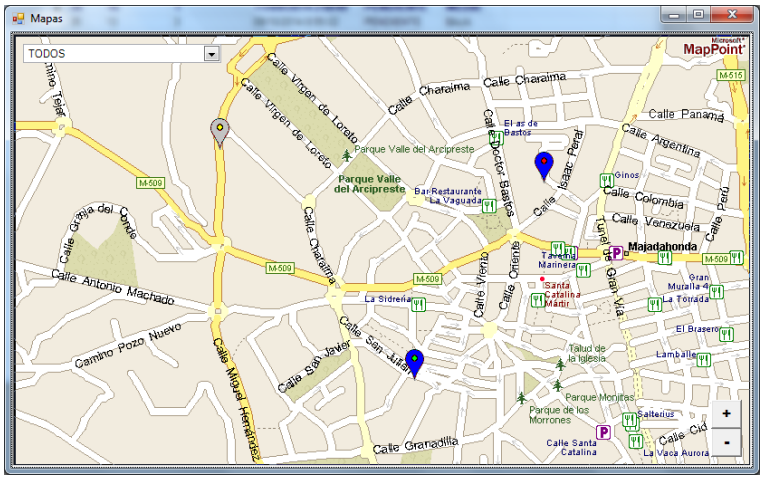


Figura 25. Aplicación cliente .Net: Mapa de avisos pendientes de realización (marcador azul para asignados a técnico y gris para no asignados).

Se puede filtrar por técnico para ver sólo la asignación de cada técnico:

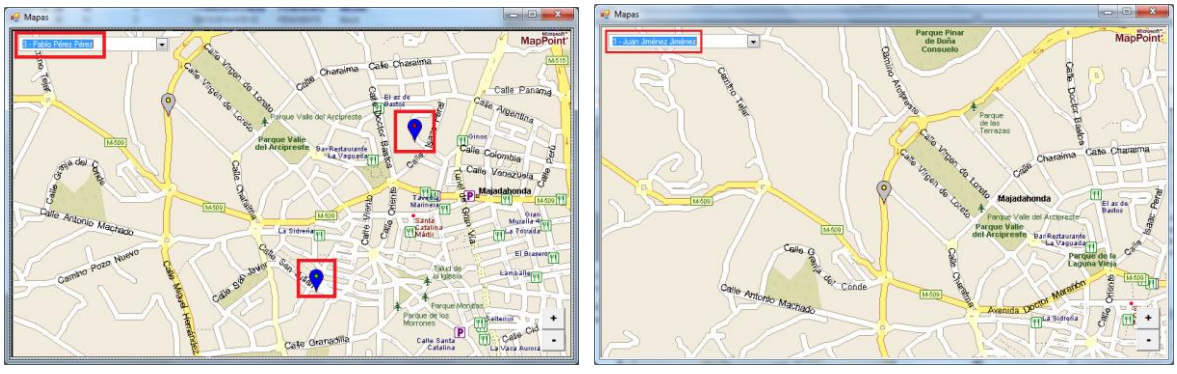


Figura 26. Aplicación cliente .Net: Mapa de avisos pendientes filtrados por técnico asignado.

En este caso, cualquiera de los dos técnicos (Pablo Pérez con dos avisos asignados y Juan Jiménez ocioso) podría ser un buen candidato a recibir el aviso, el primero por proximidad y el segundo por desocupación.

Para realizar la asignación, se selecciona el marcador gris, que muestra la ficha asociada al aviso, donde es posible asociar un técnico al aviso seleccionándolo en un formulario emergente que aparece al hacer click en el botón señalado en rojo:

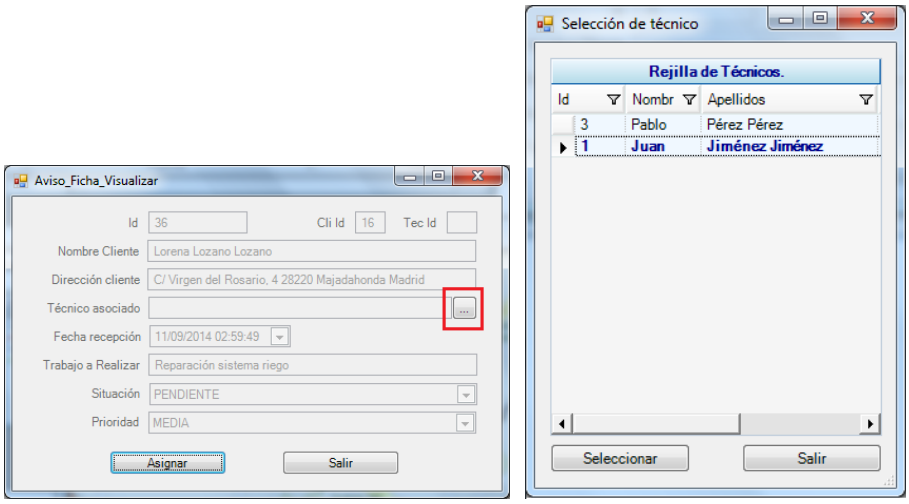


Figura 27. Aplicación cliente .Net: Asignación de aviso pendiente a técnico mediante click en el marcador gris.

Tras grabar, el aviso se muestra como asignado (marcador azul) en el mapa:

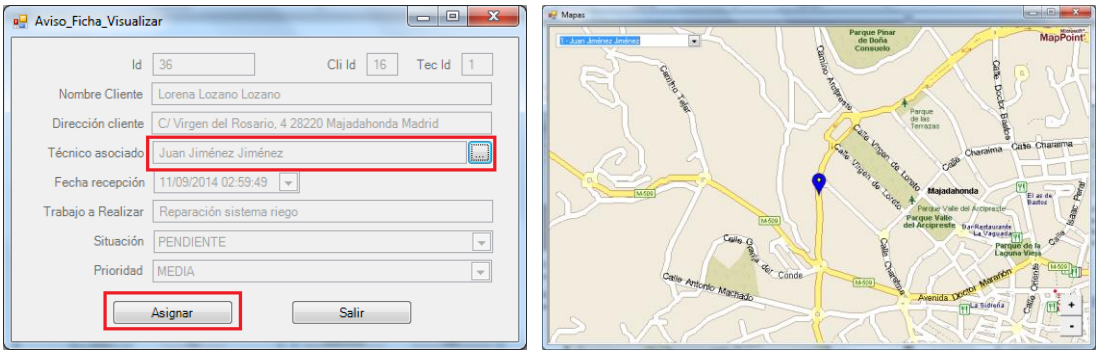


Figura 28. Aplicación cliente .Net: Grabación de aviso pendiente asociado a técnico y presentación en el mapa (marcador azul).

3.3.1.2.3 Gestión documental

La funcionalidad de esta opción es idéntica a la de gestión de documentos asociados a avisos y clientes de la opción "Clientes-Avisos-Documentos". En este caso se presenta una rejilla con todos los documentos de la aplicación, donde es posible filtrar por aviso, cliente, fecha, tipo de documento, etc.

De nuevo, en la barra lateral se presentan dos opciones: visualización de documento y eliminación del mismo.

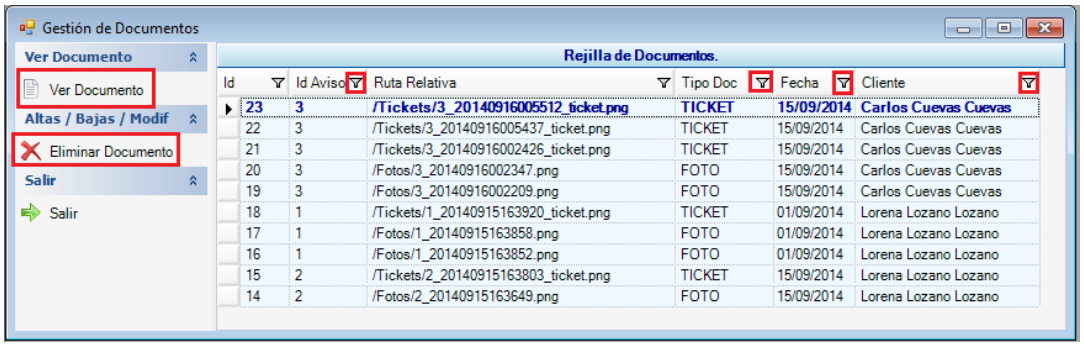


Figura 29. Aplicación cliente .Net: Rejilla de gestión documental.

3.3.1.2.4 Rutas de técnicos

Esta opción se encarga de mostrar un formulario que presenta un mapa del recorrido del técnico y de sus avisos realizados en una fecha dada:

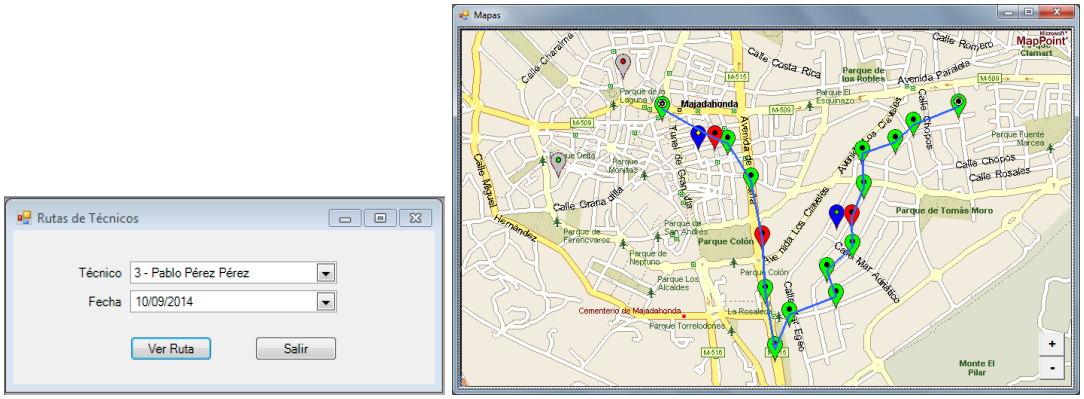


Figura 30. Aplicación cliente .Net: Presentación de ruta de técnico dada una fecha.

En el mapa se pueden ver en verde las paradas de menos de 5 minutos y en rojo las paradas superiores a ese intervalo de tiempo.

El inicio y fin de la ruta se indican con un círculo blanco en el centro del marcador:

Figura 31. Aplicación cliente .Net: Mapa de ruta de técnico. Marcadores de parada.

Además, los avisos de la ruta también se muestran en el mapa. En gris se muestran los avisos de la ruta pendientes y en azul los realizados. Los marcadores de ambos tipos de aviso contienen un círculo indicando su prioridad (rojo = alta, amarillo = media y verde = baja):

Figura 32. Aplicación cliente .Net: Mapa de ruta de técnico. Marcadores de avisos.

Este mapa permite visualizar rápidamente la equivalencia entre las paradas realizadas por los técnicos y los avisos realizados. Así puede verse, por ejemplo, que existe una parada "sospechosa" en el mapa sin ningún aviso cercano.

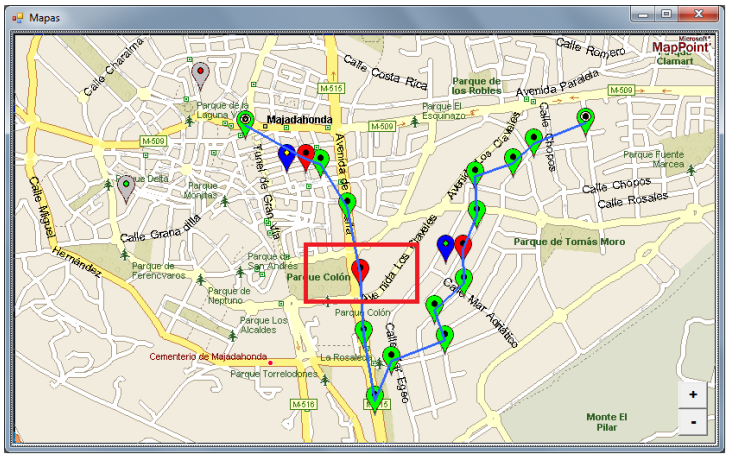


Figura 33. Aplicación cliente .Net: Mapa de ruta de técnico. Detección de paradas "sospechosas".

Aunque la parada esté asociada a un aviso, como es el caso de las otras dos, es posible comparar la información de parada con las horas de inicio y realización y con el justificante de trabajo del aviso para detectar posible absentismo del técnico. Para ello, basta con seleccionar el marcador de parada o de aviso, y la aplicación presenta un formulario emergente con la información correspondiente:

- Datos de parada:

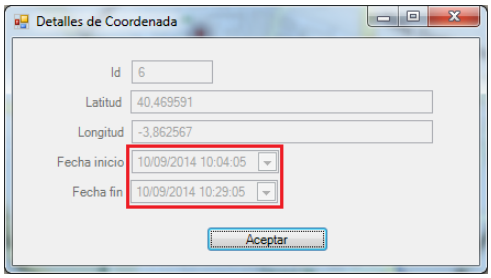


Figura 34. Aplicación cliente .Net: Mapa de ruta de técnico. Ficha de información de parada.

- Datos de aviso con opción de visualización de ticket firmado por el cliente:

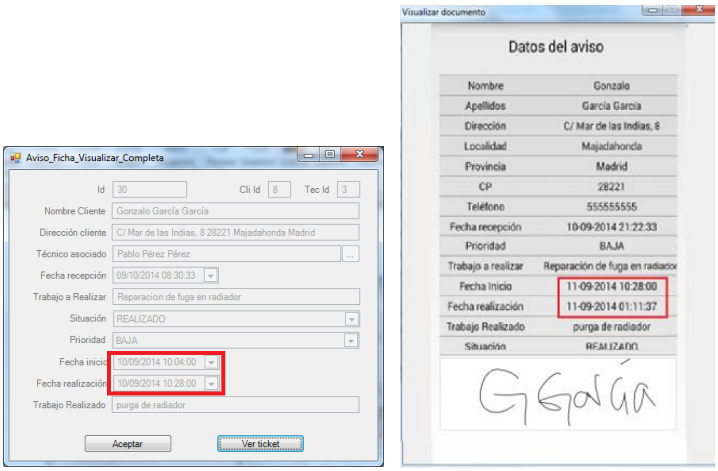


Figura 35. Aplicación cliente .Net: Mapa de ruta de técnico. Ficha de información de aviso (con opción de visualización de ticket firmado por el cliente).

3.3.1.2.5 Empleados y Técnicos

Estos formularios presentan el siguiente aspecto:

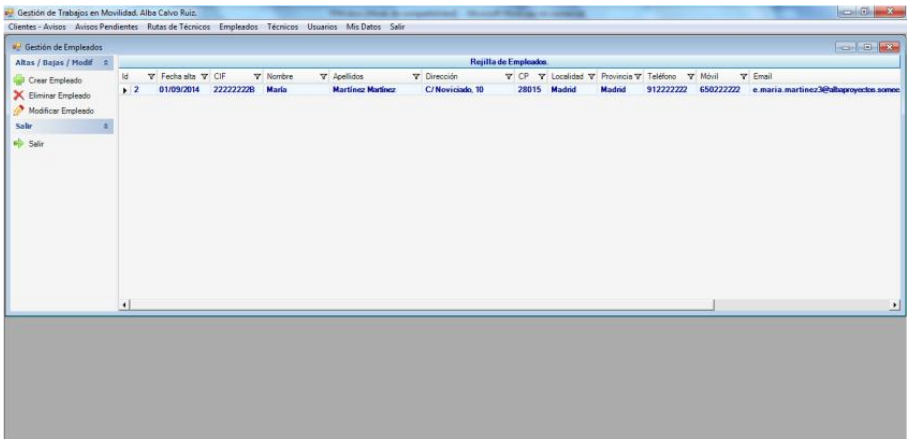


Figura 36. Aplicación cliente .Net: Rejilla de empleados.

No son ventanas de maestro/detalle, pero su funcionalidad es idéntica a la de la rejilla de clientes-avisos-documentos del primer ejemplo, por lo que no se presentarán más diagramas de funcionamiento.

Sin embargo, ambos presentan una característica que los diferencia del resto de formularios de la aplicación: su alta, baja y modificación lleva acciones asociadas.

En ambos formularios, el alta supone la creación de un usuario asociado cuyo nick se crea de la siguiente forma: "e.[nombre].[primer apellido][id empleado]" en el caso de los empleados operadores y "t.[nombre].[primer apellido][id técnico] en el caso de los técnicos. Ocurre lo mismo con el email de empleado o técnico pero añadiendo el nombre del dominio "@albaproyectos.somee.com". Además, al dar de alta un empleado operador o un técnico se genera una contraseña por defecto para la cuenta de la aplicación y para el correo que deberán cambiar más tarde: "eGTM1234" en el caso de los operadores y "tGTM1234" en el caso de los técnicos.

A continuación se ilustra el ejemplo de un alta de empleado operador:

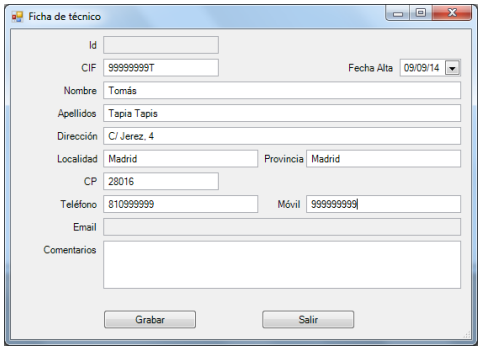


Figura 37. Aplicación cliente .Net: Alta de empleado.

Al igual que en el alta de avisos, se presenta una ficha con campos a rellenar y, tras grabarla, los cambios se ven reflejados en la rejilla de empleados:

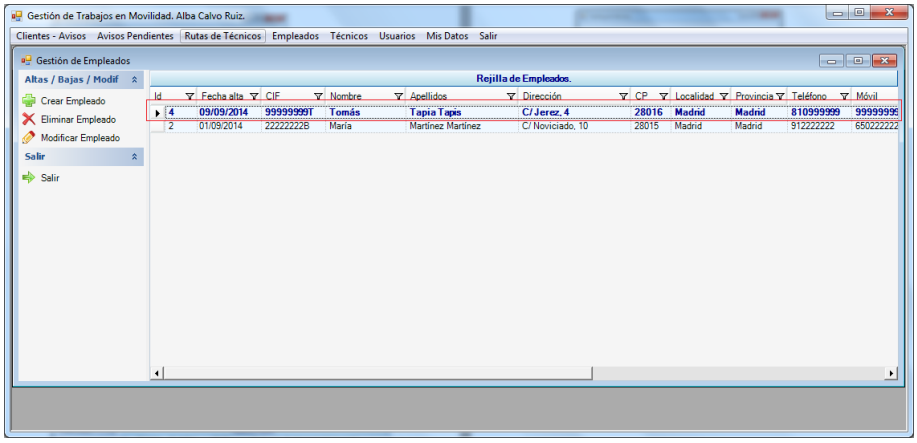


Figura 38. Aplicación cliente .Net: Rejilla de empleados con cambios reflejados.

Pero además se crea un usuario asociado, como se puede observar en el formulario "Usuarios".

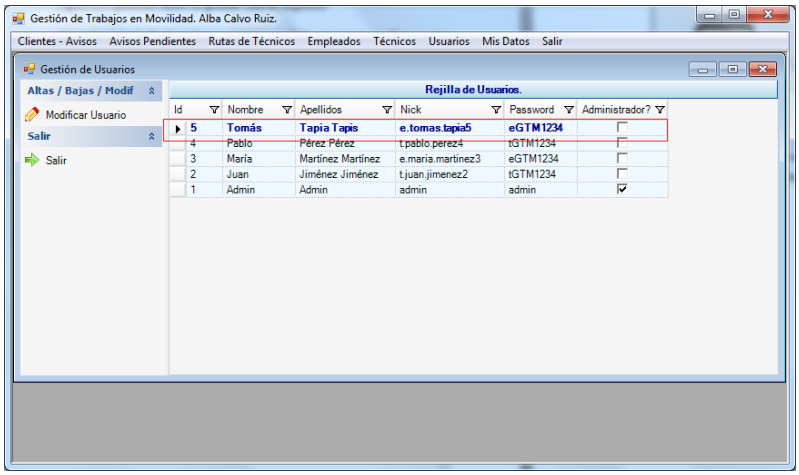


Figura 39. Aplicación cliente .Net: Rejilla de usuario con cambios reflejados.

La funcionalidad de modificación y eliminación de un empleado es muy similar y supone la modificación o eliminación del usuario asociado.

3.3.1.2.6 Usuarios

Como se puede observar en la figura siguiente, la única acción posible de este formulario es la modificación del usuario, concretamente del campo contraseña.

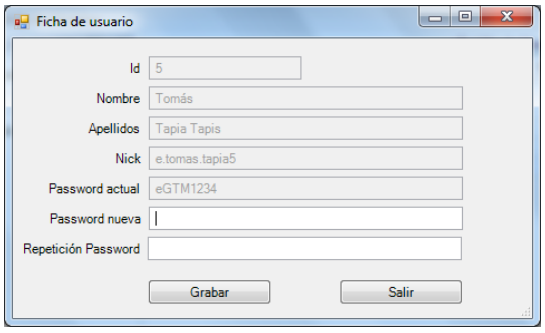


Figura 40. Aplicación cliente .Net: Modificación de usuario.

3.3.1.2.7 Mis datos

Este formulario es común a administrador y empleado (en el caso de empleados los campos nombre y apellidos se encuentran inhabilitados).

Esta ventana permite cambiar los datos del administrador en caso de que se relegue esta tarea en otra persona. También permite cambiar la contraseña.

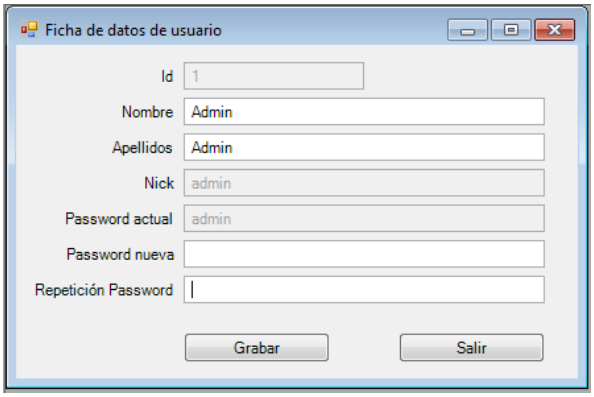


Figura 41. Aplicación cliente .Net: Modificación de datos de usuario.

3.3.1.2.8 Logout

La selección de esta opción finaliza la sesión de usuario.

3.3.2 Aplicación para terminales en movilidad

El proyecto incluye una aplicación de gestión de avisos y geolocalización para dispositivos móviles con sistema operativo Android.

El funcionamiento de la misma se resume en el siguiente diagrama de estados:

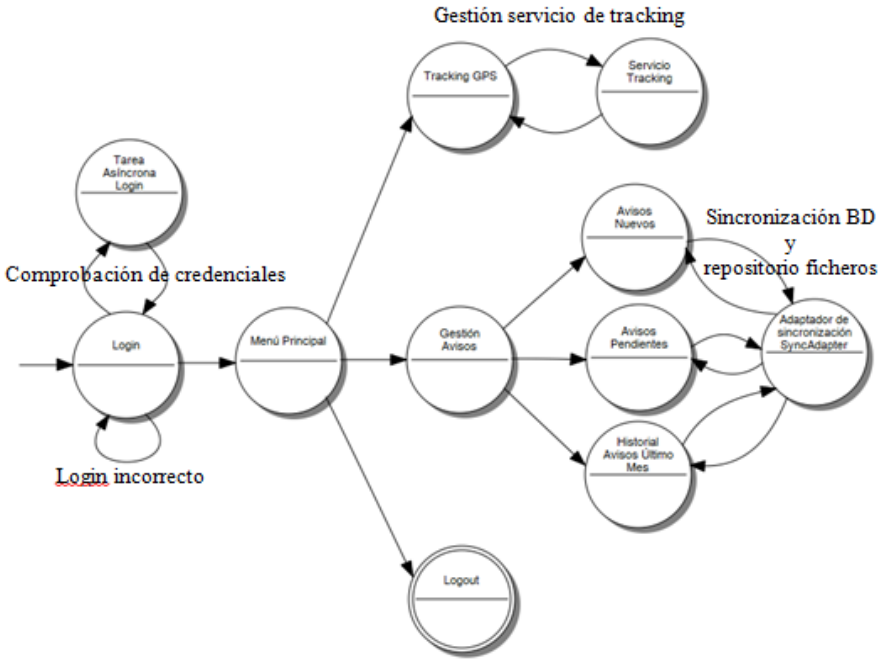


Figura 42. Aplicación cliente Android: Máquina de estados.

A continuación se muestran las pantallas de la aplicación:

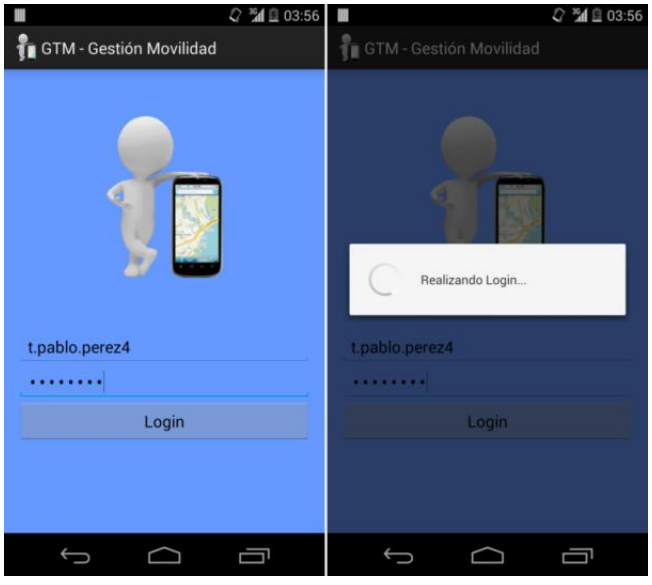


Figura 43. Aplicación cliente Android: Login.

Inicialmente se muestra la pantalla de login, donde el técnico debe autentificarse.

Si la autenticación es correcta se accede al menú principal de la aplicación, que consta de tres opciones:

Figura 44. Aplicación cliente Android: Menú principal.

3.3.2.1 Tracking GPS

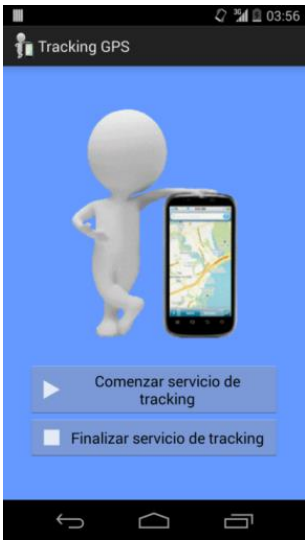


Figura 45. Aplicación cliente Android: Menú de tracking GPS.

El menú de tracking consta de dos botones para iniciar y finalizar el servicio de seguimiento respectivamente.

Los técnicos deben iniciar la aplicación al comienzo de su turno de trabajo para enviar su posición cada minuto o cada vez que recorran 100 metros y mantenerla activa durante el transcurso de su actividad. Esto activa un servicio de localización que sigue activo aunque la aplicación principal pase a segundo plano o incluso sea cerrada por el planificador de procesos, por lo que el técnico podrá hacer un uso normal de su teléfono.

La interfaz del menú es muy sencilla y consta de dos botones:

- Comenzar servicio de tracking: inicia un servicio de geolocalización en un hilo diferente al de la aplicación principal.

  La aplicación detecta la habilitación del GPS y al pulsar el botón "Comenzar el servicio de tracking", el servicio sólo comenzará si el GPS está habilitado. De lo contrario se notificará al usuario que debe activar el GPS:

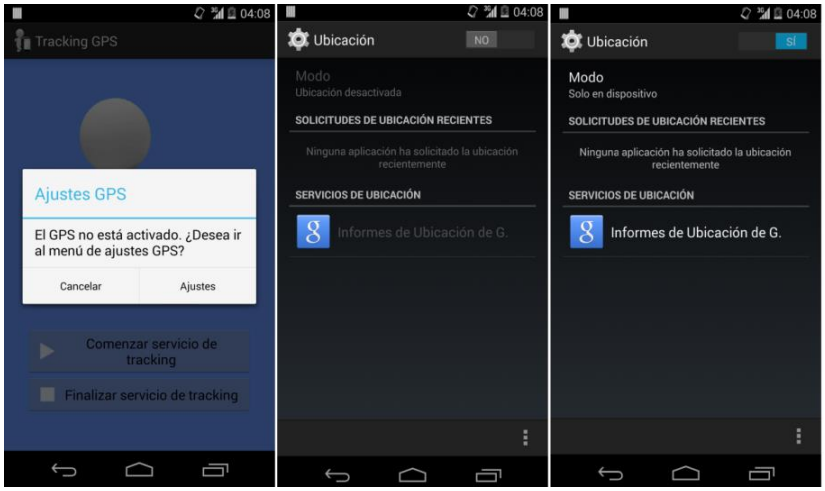


Figura 46. Aplicación cliente Android: Detección de habilitación del GPS del dispositivo.

Si todo va bien, aparecerá en la barra superior un icono que indica que se está localizando la posición del usuario vía GPS, que debe desparecer al finalizar el servicio:

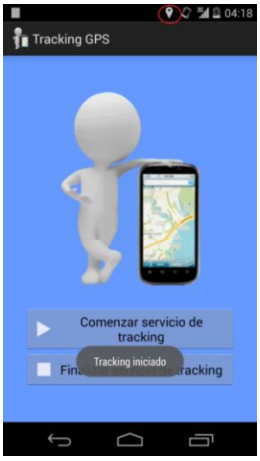


Figura 47. Aplicación cliente Android: Inicio del servicio de tracking GPS.

- Finalizar servicio de tracking: detiene el servicio de geolocalización.

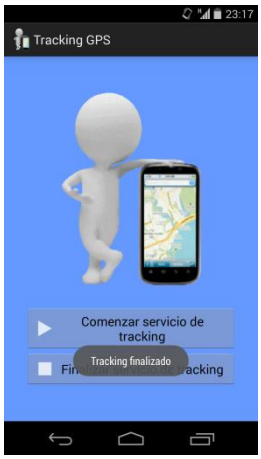


Figura 48. Aplicación cliente Android: Finalización del servicio de tracking GPS.

3.3.2.2 Gestión de avisos

Al seleccionar la opción de gestión de avisos se muestra el siguiente menú:

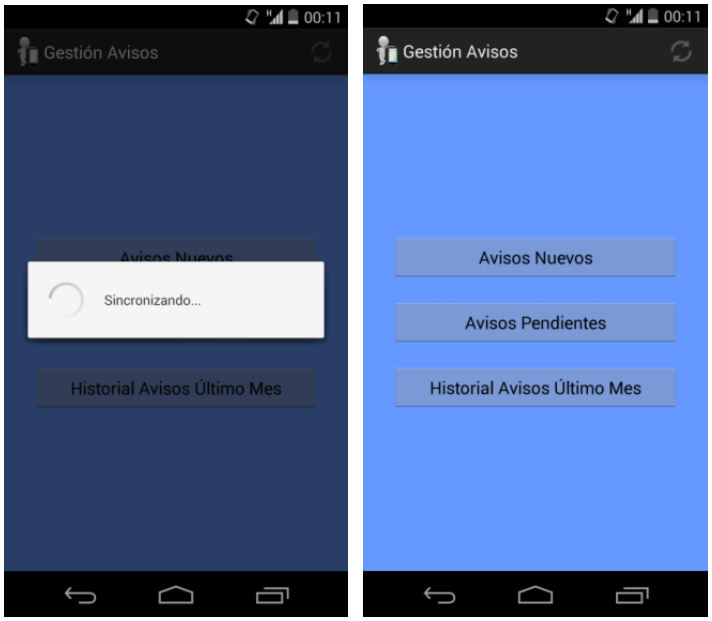


Figura 49. Aplicación cliente Android: Menú de gestión de avisos.

Inicialmente se realiza la sincronización automática con el servidor Web para descargar la información de los nuevos avisos asignados al técnico.

Existen tres opciones:

- Consulta de avisos nuevos no leídos por el técnico.
- Cumplimentación de avisos pendientes leídos en el paso anterior.
- Consulta de historial de avisos realizados en el último mes.

3.3.2.2.1 Avisos Nuevos

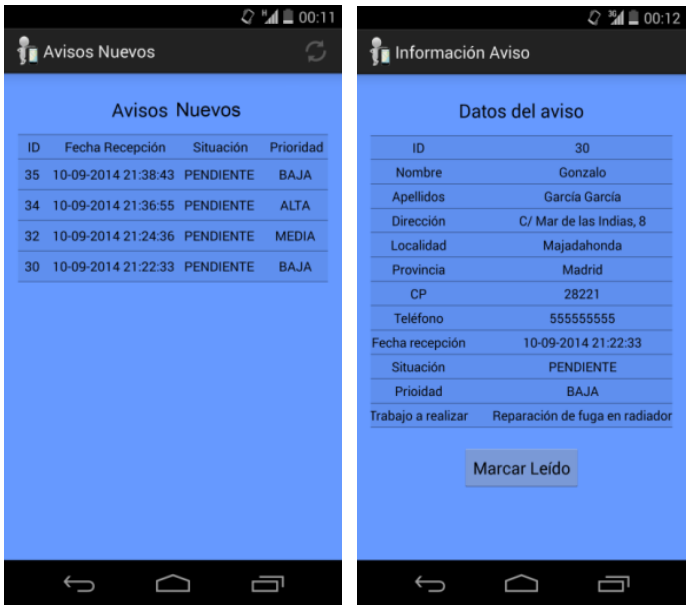


Figura 50. Aplicación cliente Android: Menú de avisos nuevos y detalle de los datos de un aviso nuevo.

En este menú se muestran los avisos recién notificados al técnico que éste aún no ha leído. Se muestra una breve información de la fecha de recepción y grado de importancia de cada aviso. Si el técnico selecciona cualquiera de ellos, se muestra información adicional. Tras esta revisión de la información, el técnico debe marcar el aviso como leído, y éste pasará a formar parte de sus avisos pendientes y desaparecerá del menú de nuevos avisos:

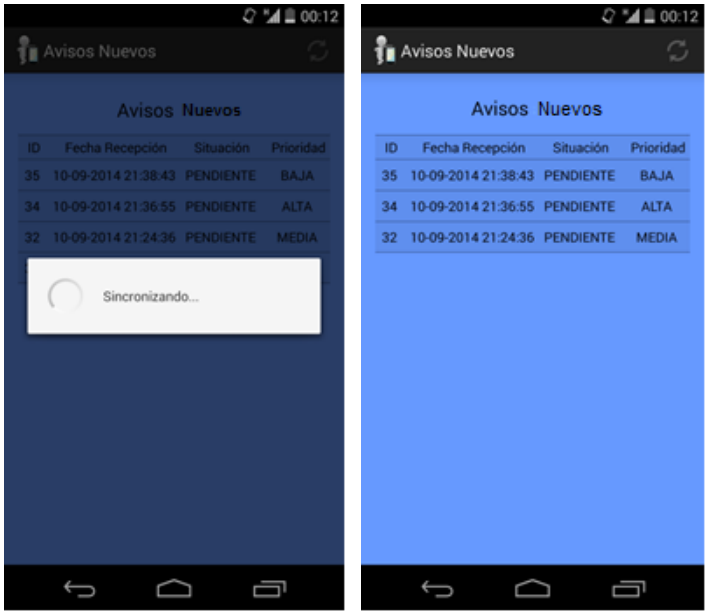


Figura 51. Aplicación cliente Android: Sincronización del aviso con la base de datos Web tras marcarlo como leído.

3.3.2.2.2 Avisos Pendientes

Los avisos pendientes de realización pueden cumplimentarse en el menú de avisos pendientes:

Figura 52. Aplicación cliente Android: Menú de avisos pendientes y detalle de un aviso pendiente.

El ejemplo ilustra la cumplimentación de un aviso, introduciendo manualmente la hora de inicio y la descripción del trabajo realizado, así como el estado de realización del aviso (REALIZADO / ANULADO).

El técnico también dispone de la opción de incluir imágenes relativas al aviso:

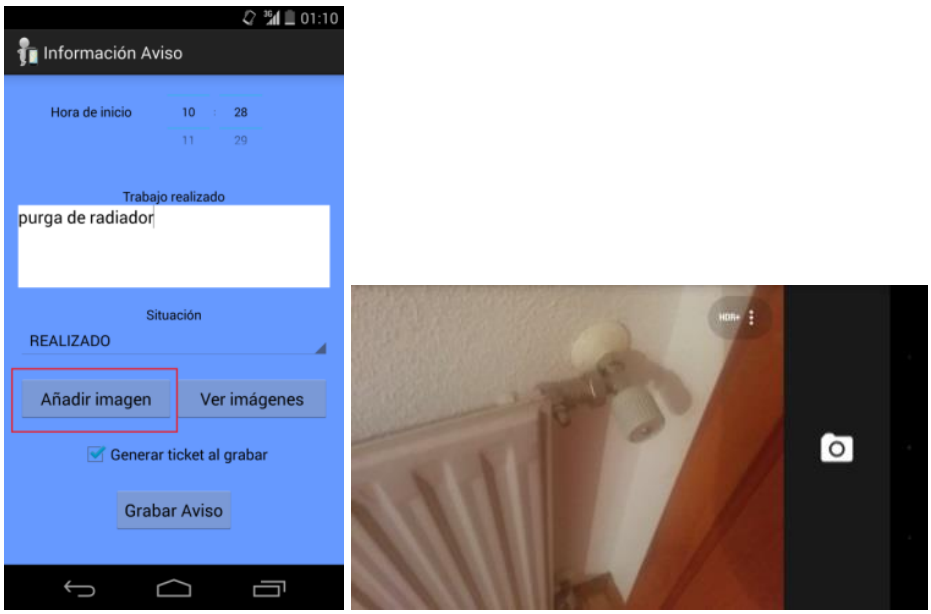


Figura 53. Aplicación cliente Android: Adición de imágenes al cumplimentar un aviso pendiente.

El técnico puede visualizar y eliminar imágenes asociadas al aviso de la siguiente forma:

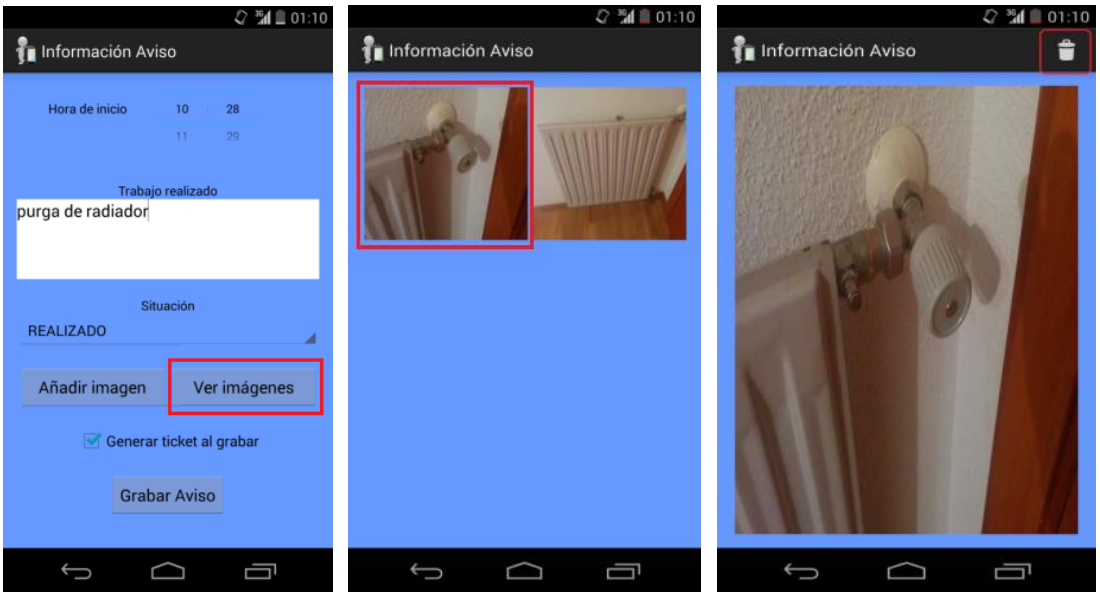


Figura 54. Aplicación cliente Android: Visualización de imágenes añadidas (con posibilidad de eliminación) al cumplimentar un aviso.

Por último, el técnico dispone de la opción de generar un justificante de trabajo a firmar por el cliente al grabar el aviso marcando la opción "Generar ticket al grabar" (una vez grabado el aviso, también es posible generar el justificante desde el historial de avisos seleccionando el aviso deseado):

Figura 55. Aplicación cliente Android: Generación de un ticket firmado por el cliente al grabar el aviso cumplimentado.

3.3.2.2.3 Historial de avisos realizados el último mes

En esta opción pueden verse los detalles de los avisos realizados durante el último mes:

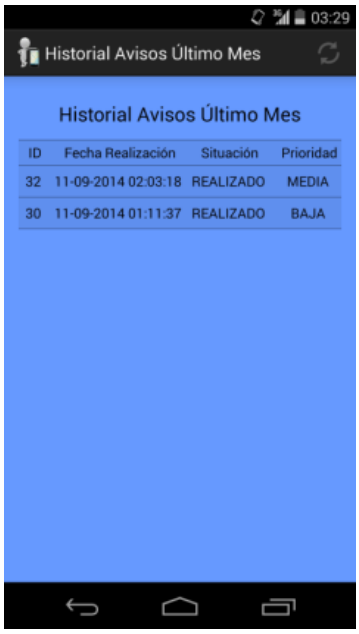


Figura 56. Aplicación cliente Android: Menú de historial de avisos realizados en el último mes.

Si se selecciona un aviso, se muestran sus detalles, así como dos botones de visualización de imágenes y justificante asociados respectivamente:

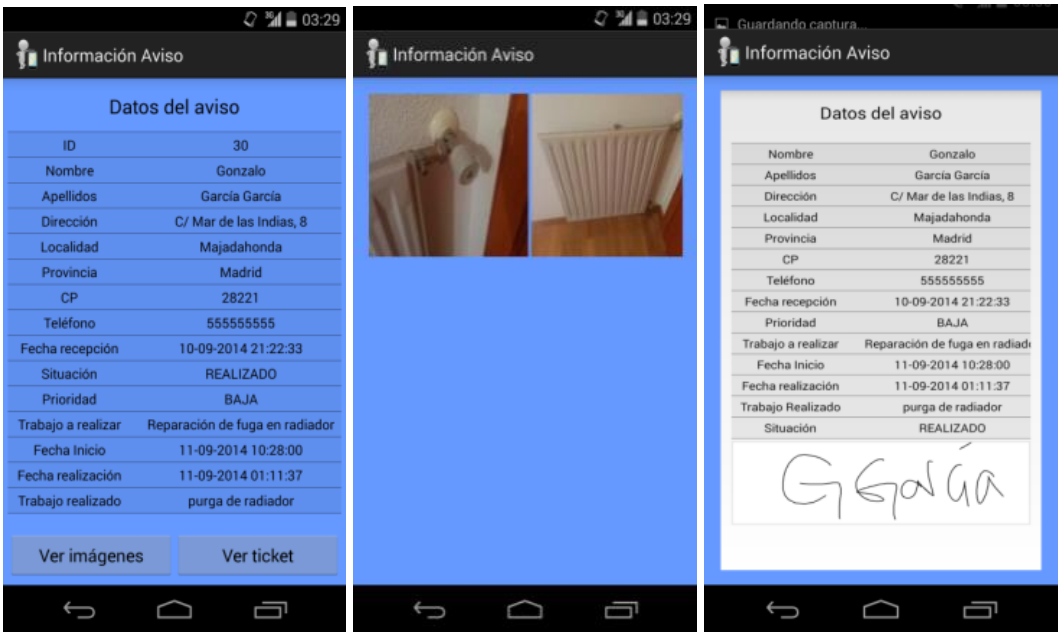


Figura 57. Aplicación cliente Android: Detalle de un aviso del historial, incluyendo visualización de imágenes asociadas y de ticket firmado por el cliente (en caso de tenerlos).

En caso de que el justificante no haya sido generado, se dispone de una opción para la generación del mismo:

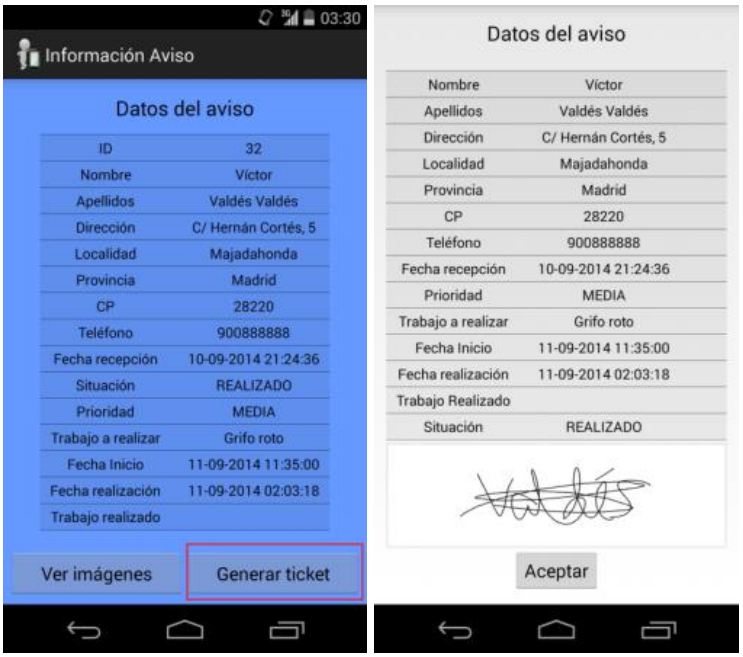


Figura 58. Aplicación cliente Android: Generación de ticket firmado por el cliente desde el detalle de un aviso del historial.

3.3.2.3 Logout

Esta opción detiene el servicio de geolocalización en caso de que esté activo y cierra la sesión de usuario.

(Para finalizar la aplicación no es necesario hacer Logout, sino que basta con pulsar la tecla Home o back del teléfono. Esta forma de salir de la aplicación es recomendable, ya que así el usuario no tiene que autentificarse cada vez que utiliza la aplicación).

3.4 Ejemplo de flujo de la aplicación

Aunque en los apartados anteriores se ha explicado en detalle el funcionamiento de los distintos subsistemas de la aplicación, dada la complejidad del ciclo de vida de un aviso, se presentará un ejemplo del mismo:

1. El empleado operador recibe una llamada y da de alta el aviso correspondiente en la aplicación cliente .Net de su PC. El operador no asigna técnico al aviso para poder asignarlo luego desde la pestaña "Avisos Pendientes" atendiendo a la creación de rutas óptimas con avisos en la misma área:

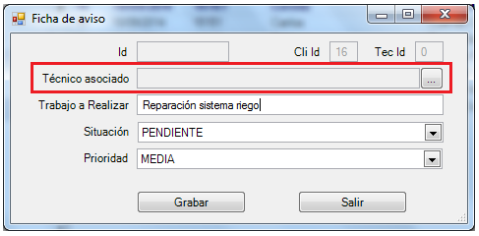


Figura 59. Aplicación cliente .Net: Alta de aviso sin técnico asociado.

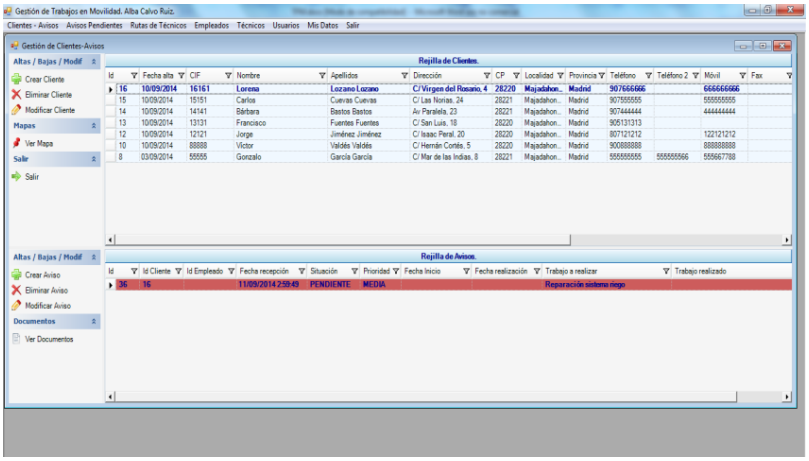


Figura 60. Aplicación cliente .Net: Avisos sin técnico asociado indicados en rojo en la rejilla Clientes-Avisos-Documentos.

2. El operador accede a la pestaña de "Avisos Pendientes" para realizar la asignación del aviso:

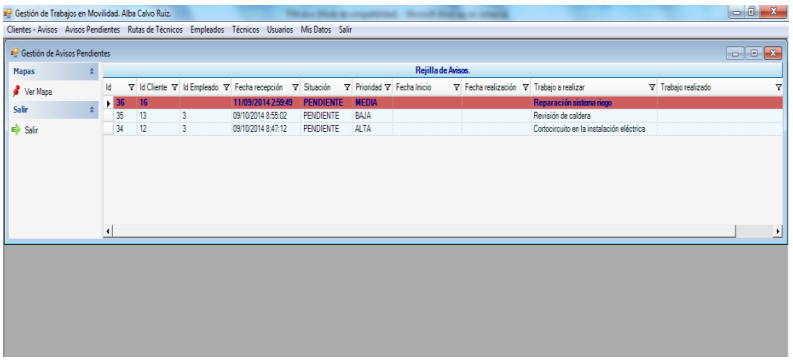


Figura 61. Aplicación cliente .Net: Rejilla de avisos pendientes.

El operador selecciona la opción de visualización del mapa y compara la localización del nuevo aviso recién dado de alta sin asignar (en gris) con las rutas diarias asignadas a cada uno de sus técnicos hasta el momento:

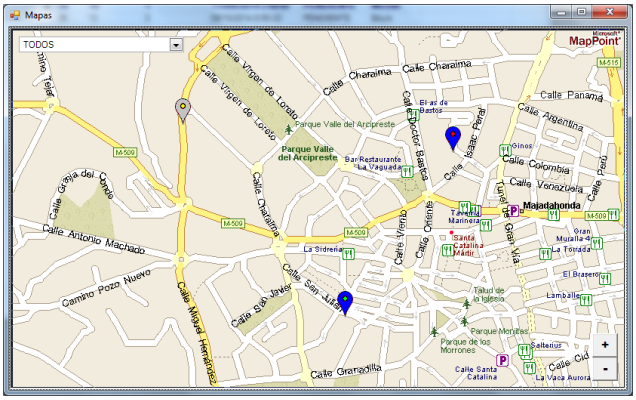


Figura 62. Aplicación cliente .Net: Mapa de avisos pendientes de realización (marcador azul para asignados a técnico y gris para no asignados).

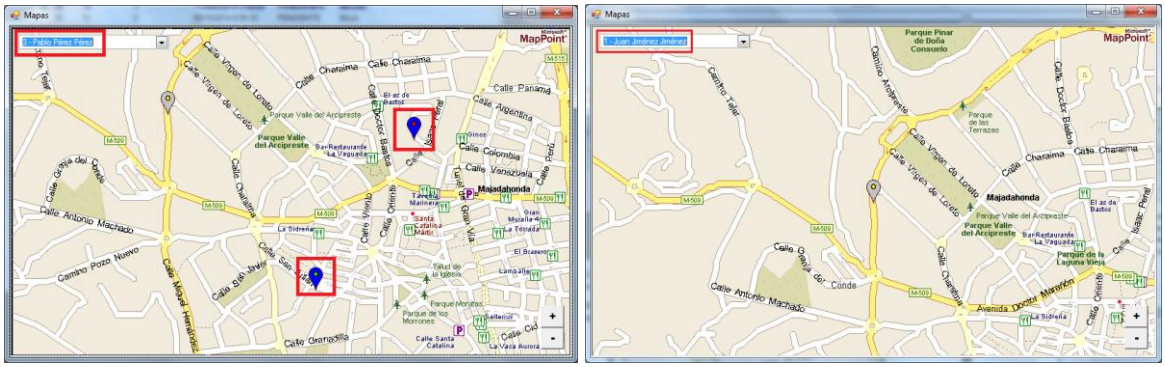


Figura 63. Aplicación cliente .Net: Mapa de avisos pendientes filtrados por técnico asignado.

3. El operador asigna el aviso al técnico Juan Jiménez:

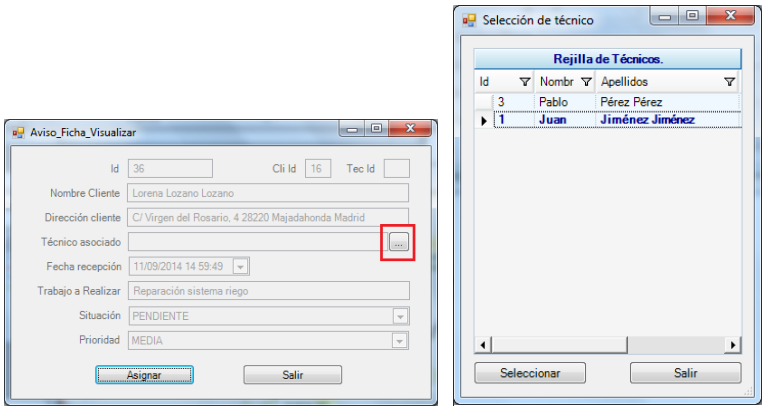


Figura 64. Aplicación cliente .Net: Asignación de aviso pendiente a técnico mediante click en el marcador gris.

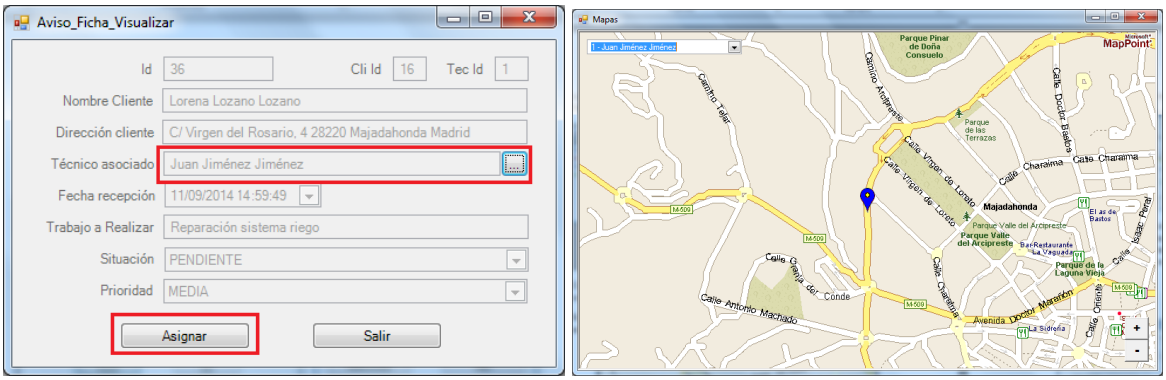


Figura 65. Aplicación cliente .Net: Grabación de aviso pendiente asociado a técnico y presentación en el mapa (marcador azul).

4. El técnico recibe una notificación en su correo electrónico y procede a autenticarse en la aplicación terminal Android para consultarlo y marcarlo como leído:

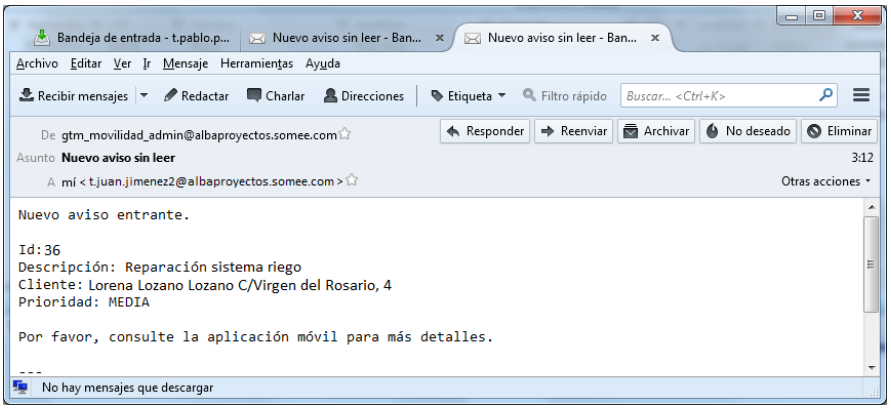


Figura 66. Mensaje de notificación de recepción de aviso.

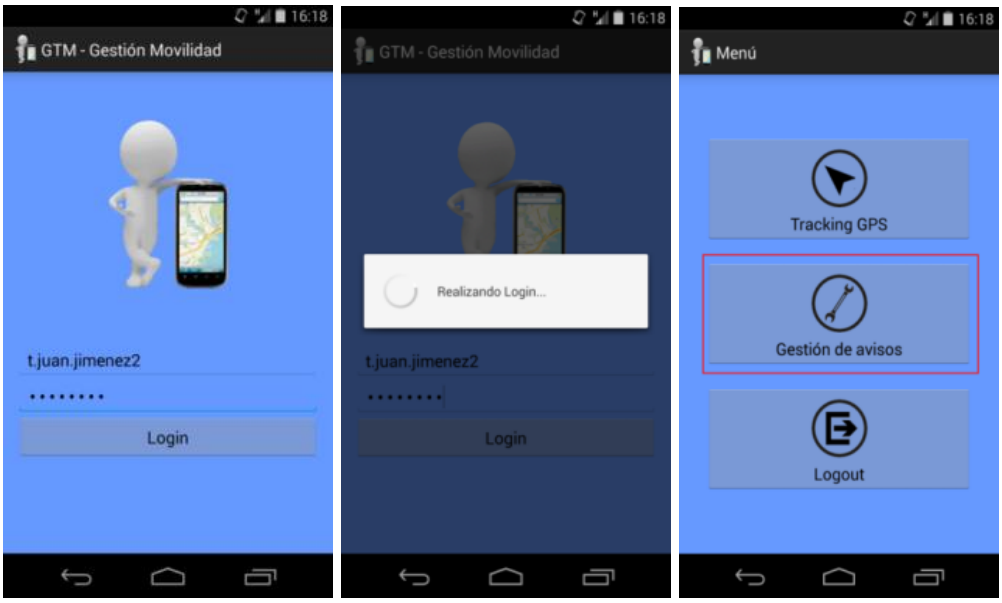


Figura 67. Aplicación cliente Android: Autenticación de técnico y acceso al menú de gestión de avisos.

El técnico marca el aviso nuevo como leído:

Figura 68. Aplicación cliente Android: Marcado de aviso nuevo como leído.

5. El técnico acude al domicilio y cumplimenta el aviso, incluyendo imágenes:

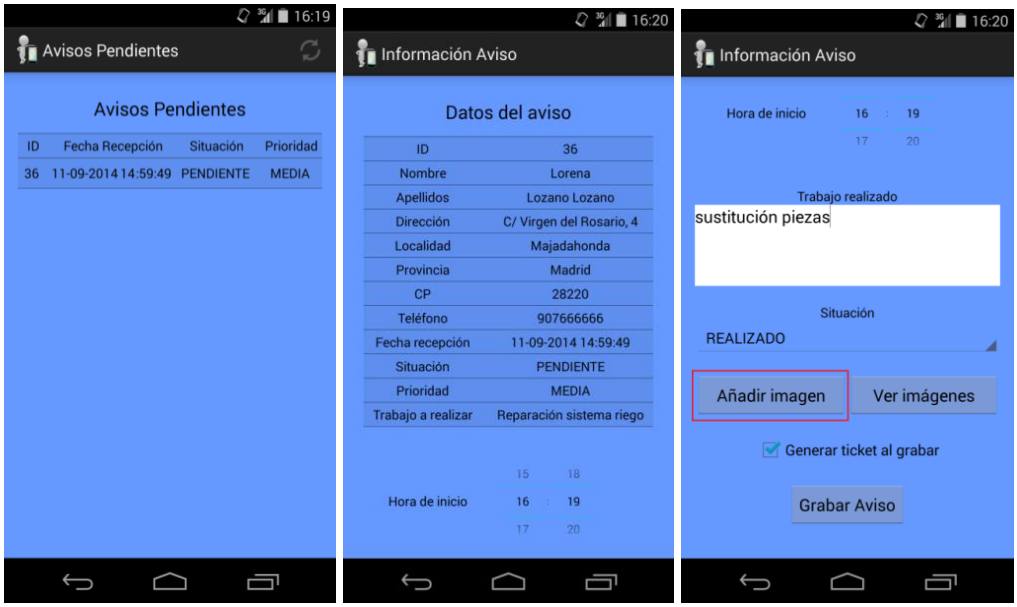


Figura 69. Aplicación cliente Android: Cumplimentación de aviso pendiente.

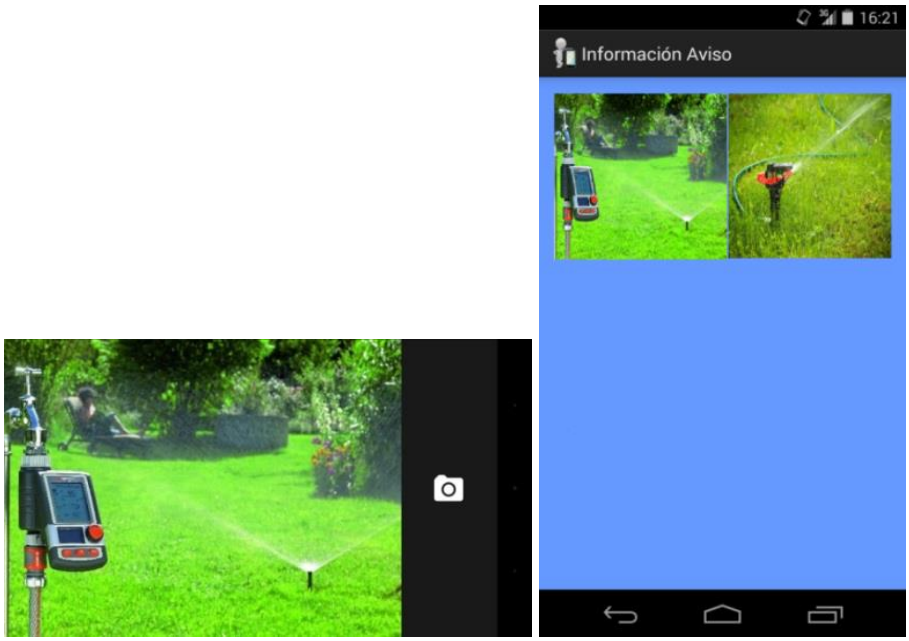


Figura 70. Aplicación cliente Android: Cumplimentación de aviso pendiente. Adición de imágenes.

El técnico graba el aviso marcando la opción de generación de ticket firmado por el cliente:

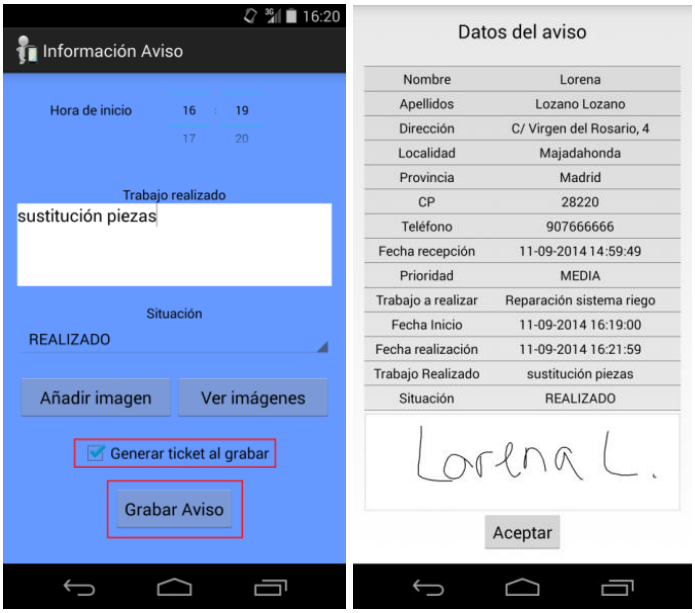


Figura 71. Aplicación cliente Android: Grabación de aviso con opción de generación de ticket firmado por el cliente habilitada.

6. El técnico comprueba el historial de avisos del último mes:

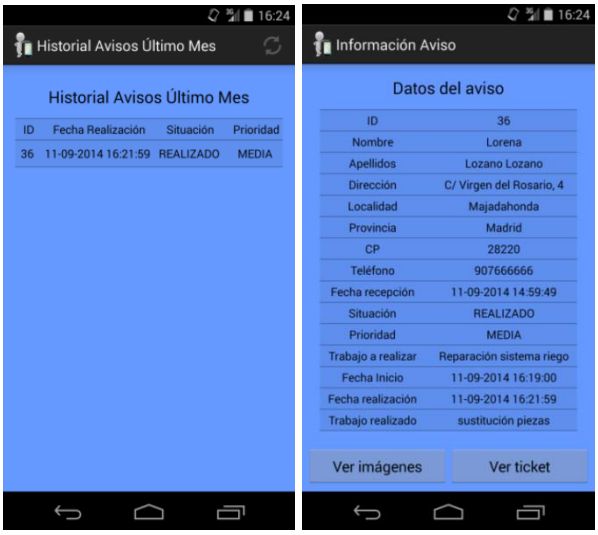


Figura 72. Aplicación cliente Android: Menú de historial de avisos del último mes y detalle del aviso realizado.

El técnico comprueba las imágenes asociadas al aviso y el ticket firmado por el cliente:

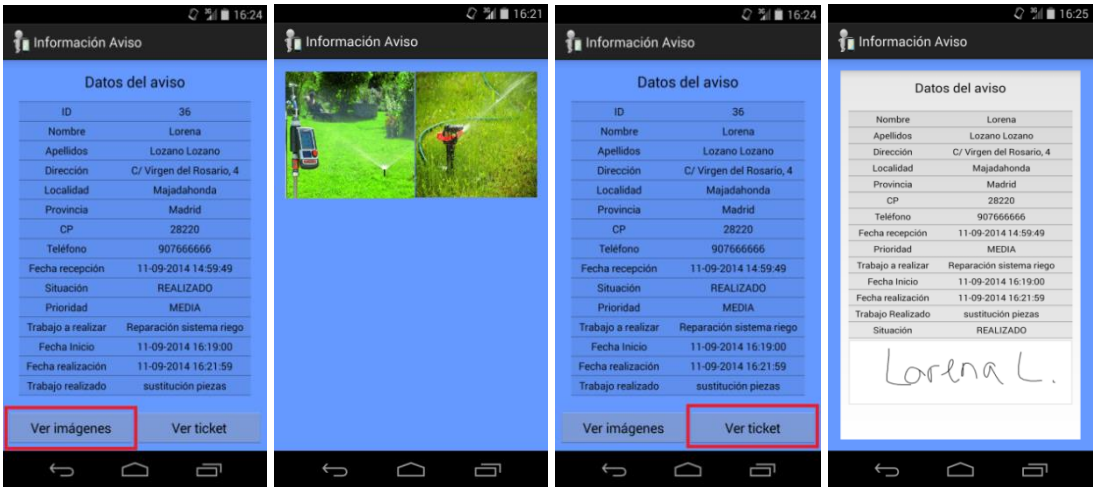


Figura 73. Aplicación cliente Android: Visualización de imágenes y ticket asociados al aviso.

3.5 Componentes del proyecto. Análisis de tecnologías

El proyecto GTM (Gestión de Trabajos en Movilidad) v2.0 incluye los siguientes componentes fundamentales:

- Servidor Web de base de datos SQLServer.
- Aplicación cliente corporativa instalada en PCs de usuarios.
- Aplicación cliente instalada en los terminales en movilidad de los técnicos.

La consulta y almacenamiento de información en base de datos Web se realiza mediante el ORM (Object-Relational Mapping) Entity Framework en el caso de la aplicación cliente corporativa y a través de una API Web en el caso de la aplicación en terminales en movilidad de los técnicos. Ambos se explicarán en detalle más adelante.

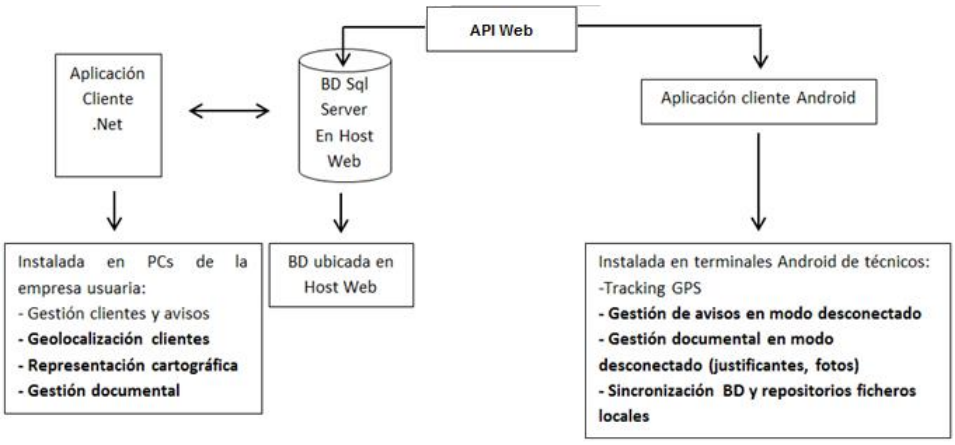


Figura 74. Esquema GTM v2.0.

Para la elección de la tecnología de cada uno de los componentes y del servidor de base de datos y la API se han tenido en cuenta las opciones predominantes en el mercado actual identificadas en el capítulo 2 "Estado del arte".

A continuación se detallan las tecnologías seleccionadas, así como el criterio utilizado durante el proceso de evaluación:

3.5.1 Base de datos SQLServer

Se ha descartado la opción de MySQL debido a las incertidumbres derivadas de su adquisición por Oracle. En cuanto a la alternativa MariaDB, aunque en rápido crecimiento, se ha desestimado por no ser aún un producto consolidado.

Aunque se trata de un producto de pago que es necesario licenciar, se ha optado por SQLServer frente a PostgreSQL teniendo en cuenta las ventajas que ofrece en cuanto a consolidación y herramientas de mantenimiento y productividad más amigables, así como por su rápida curva de aprendizaje.

Su elección ha permitido la utilización del ORM (Object-Relational Mapping) Microsoft Entity Framework de alta productividad para permitir la interoperabilidad entre la aplicación cliente .Net y la base de datos.
Para empresas de pequeño tamaño sería posible utilizar en un principio el producto SQLExpress de carácter gratuito que actualmente cuenta con un límite de almacenamiento de 10 GB.[22]

3.5.2 Aplicación cliente .Net con formularios Windows Forms

La aplicación da soporte a la gestión de la empresa, manteniendo los datos almacenados en la base de datos a través de una interfaz de usuario.

Además de la opción de aplicación cliente se ha considerado la posibilidad de desarrollo de una aplicación Web. Durante el proceso de evaluación se han tenido en cuenta las siguientes características de cada solución: [8] [9]

[22] http://www.microsoft.com/es-es/download/details.aspx?id=35579

- Aplicación Web:

  En esta opción el usuario final interactúa por medio de un navegador Web con la aplicación localizada en el servidor, donde residen los datos y la lógica de la aplicación.

Figura 75. Arquitectura de una aplicación Web.

  La aplicación Web presenta las ventajas inherentes a este tipo de aplicación como son la actualización instantánea sin necesidad de actuar en los PCs del usuario y el acceso desde cualquier ubicación utilizando un terminal dotado de navegador. Esta ventaja es determinante para implantaciones en grandes organizaciones con un elevado número de usuarios.

  En el caso concreto del proyecto GTM, el posible soporte futuro de la actividad completa de una empresa SAT puede conducir a una interfaz de usuario compleja. En esta situación la productividad en el desarrollo sería superior en el entorno de aplicación cliente.

- Aplicación Cliente:

  En esta opción la lógica se desplaza al cliente, que asume todas las funcionalidades de la aplicación, dejando al servidor la gestión de las transacciones contra la base de datos.

Figura 76. Arquitectura de una aplicación cliente.

  Esta opción presenta la ventaja de una alta productividad a la hora de construir interfaces con un elevado número de formularios o con un comportamiento sofisticado de los mismos (comportamiento MDI donde las ventanas hijas residen bajo una única ventana padre que se encarga de mantener el estado de las mismas[23], navegación entre formularios, ventanas maestro/detalle, etc.).

[23] http://en.wikipedia.org/wiki/Multiple_document_interface

El principal inconveniente es la necesidad de actuar en el PC de cada usuario para instalar y actualizar la aplicación. Esto puede ser un gran problema en grandes organizaciones con un elevado número de usuarios.

En la comparación de tecnologías se ha tenido en cuenta la posible evolución futura de la aplicación con vistas al soporte de la gestión completa de la actividad SAT de una empresa con la proliferación y complejidad de formularios que ello podría suponer. Por otra parte, se ha tenido en cuenta que el proyecto va dirigido a empresas de tamaño pequeño o mediano más que a grandes estructuras lo cual limita el problema de la implantación de nuevas versiones en múltiples PCs de usuario.

Teniendo en cuenta las consideraciones anteriores, se ha optado por la opción de aplicación cliente.

Al realizar esta selección se ha pensado en un escenario de trabajo consistente en una aplicación cliente atacando a un servidor de base de datos ubicado en las propias dependencias de la empresa. La implantación del escenario de pruebas del proyecto no se ajusta a esta situación, ya que la base de datos SQLServer se encuentra ubicada en un hosting Web. Esta configuración provisional obedece a la necesidad de poder realizar pruebas y demostraciones con terminales atacando a un servidor de base de datos publicado en la Web.

Como lenguaje de desarrollo para la aplicación cliente se ha optado por C# sobre la plataforma .Net. Se ha tenido en cuenta que la plataforma de desarrollo de Microsoft tiene una fuerte implantación y orientación a aplicaciones cliente.

Dentro del entorno .Net, se ha optado por Windows Forms frente a Windows PresentationFoundation (WPF) a pesar de las avanzadas prestaciones de esta última opción como formularios escalables, separación entre diseño y programación, etc. En la decisión ha prevalecido la alta productividad de Windows Forms frente a las elevadas prestaciones de WPF, sobre todo teniendo en cuenta que se trata de una aplicación de uso interno.

3.5.3 Aplicación cliente de gestión de avisos y tracking Android

Todas las plataformas consideradas en el capítulo 2 “Estado del arte” incorporan funcionalidad GPS, por lo que este aspecto no ha sido decisivo a la hora de seleccionar uno de ellos.

De acuerdo con las características expuestas en el capítulo (líder en prestaciones, alta implantación, sistema operativo de libre distribución, lenguaje de desarrollo Java), se ha seleccionado la plataforma Android como la más idónea para el desarrollo del proyecto, aunque se deja abierta la posible ampliación a otros sistemas operativos móviles.

3.5.4 API Web REST en PHP

Para la realización de una API de servicios Web se ha elegido la tecnología REST debido a su soporte para la utilización de ficheros JSON (Javascript Object Notation) como almacén de información, mucho más ligeros que los ficheros XML. Esto resulta muy útil cuando se dispone de recursos limitados como es el caso de la aplicación para los terminales en movilidad.

Además, REST presenta una curva de aprendizaje menor que SOAP y una menor dependencia de herramientas.

Como se ha expuesto anteriormente, tanto REST como SOAP son independientes del lenguaje, por lo que este factor no ha sido relevante en la decisión de REST frente a SOAP.

Una vez elegida la tecnología REST, he optado por la utilización de PHP para el desarrollo de la API de servicios Web debido a su curva de aprendizaje rápida. Además, ya contaba con una implementación en PHP de funciones de acceso a la base de datos SQL Server de la versión 1.0 de GTM.

3.5.5 Representación cartográfica con MapPoint

Se ha elegido la opción de Microsoft MapPoint para la representación cartográfica ya que, aunque es necesario adquirir una licencia para cada equipo usuario, se trata de una solución permanente que sólo hay que pagar una vez. La opción de Google Maps, además de ser mucho más cara, supone un pago anual por la utilización del servicio y no resulta rentable.

Además, MapPoint incluye librerías para .Net, por lo que la integración con la aplicación de escritorio es muy sencilla. Al estar instalado en el equipo local, toda la cartografía es accesible sin necesidad de conexión a Internet, presentando una mayor velocidad de operación y una menor latencia que los servicios Web de Google.

Por último, MapPoint no presenta un límite de solicitudes diarias, pudiendo resolver muchas más direcciones que Google Maps.

4 Implementación

A continuación se analizará en detalle la estructura y el comportamiento de cada una de las partes del proyecto GTM desde el punto de vista del desarrollador:

- Base de datos SQL Server
- Aplicación cliente de gestión C#
- Aplicación cliente Android
- API Web para la interoperabilidad con la base de datos

4.1 Base de datos

4.1.1 Tecnologías empleadas

Para realizar la configuración, gestión y administración de la base de datos se ha utilizado la herramienta SQL Server Management Studio 2012, un sistema basado en el modelo relacional con soporte para lenguajes ANSI SQL.

Esta herramienta permite la creación de la base de datos en un servidor SQL Server remoto y posibilita la gestión de la misma.

4.2 Aplicación cliente de gestión y representación cartográfica

4.2.1 Tecnologías empleadas

Para codificar la aplicación cliente se ha empleado el entorno de desarrollo Microsoft Visual Studio 2010 empleando el lenguaje de programación C# y el framework .Net.

Para el desarrollo de la interfaz gráfica se ha empleado Infragistics, una herramienta de desarrollo de interfaces de usuario para ASP.Net que incluye controles para su uso dentro de formularios.

4.2.2 Estructura

La aplicación consta de varias capas:

- Modelo de datos (ModeloDTO)
- Interfaz (MovilidadPresentacion)
- Capa de negocio (Negocio)

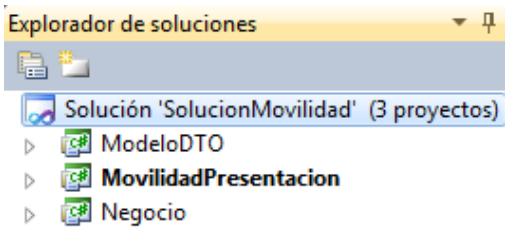


Figura 77: Capas de la aplicación cliente.

4.2.2.1 ModeloDTO y Negocio

Estas dos capas están estrechamente relacionadas entre sí y se encargan del acceso a la información de la base de datos.

Para acceder a esta información se utiliza el Entity Framework de ADO.Net, que consiste en un mapeo objeto-relacional (ORM) para el framework .Net que permite crear aplicaciones con acceso a datos programando contra un modelo conceptual de aplicación en lugar de directamente contra un esquema de almacenamiento relacional. Se consigue así disminuir la cantidad de código y simplificar el mantenimiento de la aplicación.

Para realizar el mapeo es necesario generar un modelo de base de datos. Visual Studio dispone de un asistente para crear una clase contexto (ProyectoMovilidadContext), que controla la conexión a la base de datos y el flujo de datos.

La capa de modelo de datos incluye los DTOs (Data Transfer Object), objetos que sirven para mapear cada una de las tablas de la base de datos a una clase que pueda ser utilizada en el código.

En la capa de negocio se encuentran los DAOs (Data Access Object), objetos que permiten el acceso a la base de datos de forma abstracta a través del conjunto de métodos que proporcionan. El DAO, tras una petición del controlador, rellena un DTO con los datos deseados y lo devuelve al controlador.

El uso de DAOs y DTOs permite así la separación entre la presentación y el acceso a datos y aporta al código gran flexibilidad, pues ante cualquier cambio en la forma de acceder a los datos (cambio de base de datos, obtención de datos de servicios Web, etc) el modelo no se ve afectado, simplemente sería necesario cambiar los DAOs involucrados.

A continuación se expone un ejemplo de DAO y DTO de la tabla cliente para ilustrar la explicación anterior:

DTO cliente:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ModeloDTO.Cliente
{
    public class ClienteDto : BaseDto
    {
        public ClienteDto()
        {
            fechaAlta = DateTime.Now;
            cif = "";
            nombre = "";
            apellidos = "";
            direccion = "";
            direccionLatitud = "-1";
            direccionLongitud = "-1";
            d_postal = "";
            localidad = "";
            provincia = "";
            tlf = "";
            tlf_2 = "";
            tlf_movil = "";
            fax = "";
            email = "";
            Web = "";
            comentarios = "";
        }

        public Int32 id { get; set; }
        public DateTime? fechaAlta { get; set; }
        public string cif { get; set; }
        public string nombre { get; set; }
        public string apellidos { get; set; }
        public string direccion { get; set; }
        public string direccionLatitud { get; set; }
        public string direccionLongitud { get; set; }
        public string d_postal { get; set; }
        public string localidad { get; set; }
        public string provincia { get; set; }
        public string tlf { get; set; }
        public string tlf_2 { get; set; }
        public string tlf_movil { get; set; }
        public string fax { get; set; }
        public string email { get; set; }
        public string Web { get; set; }
        public string comentarios { get; set; }
    }
}
```

DAO cliente:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Negocio.Controller
{
    public class ClienteController
    {
        public ModeloDTO.Cliente.ClienteDto nuevoDto()
        {
            return new ModeloDTO.Cliente.ClienteDto();
        }

        public ModeloDTO.Cliente.ClienteDto getDto(Int32 idCliente)
        {
            var contexto = new ProyectoMovilidadContext();
            var listaClientes = from cli in contexto.cliente
                                where cli.id == idCliente
                                select new ModeloDTO.Cliente.ClienteDto()
                                {
                                    id = cli.id,
                                    fechaAlta = cli.fecha_alta,
                                    cif = cli.cif,
                                    nombre = cli.nombre,
                                    apellidos = cli.apellidos,
                                    direccion = cli.direccion,
                                    direccionLatitud = cli.direccion_latitud,
                                    direccionLongitud = cli.direccion_longitud,
                                    d_postal = cli.d_postal,
                                    localidad = cli.localidad,
                                    provincia = cli.provincia,
                                    tlf = cli.tlf,
                                    tlf_2 = cli.tlf_2,
                                    tlf_movil = cli.tlf_movil,
                                    fax = cli.fax,
                                    email = cli.email,
                                    Web = cli.Web,
                                    comentarios = cli.comentarios
                                };

            ModeloDTO.Cliente.ClienteDto dtoCli = listaClientes.First();
            contexto.Dispose();
            return dtoCli;
        }

        ...
    }
}
```

En el DAO se incluyen todos los métodos de obtención de datos necesarios: nuevoDto, getDto (en el ejemplo superior)...

Si analizamos cualquiera de ellos en detalle podemos ver que instancian una variable contexto de la clase ProyectoMovilidadContext. Esta clase es el modelo que se ha generado a partir del asistente de Visual Studio para controlar la conexión a la base de datos y el flujo de datos. A continuación se realiza una consulta utilizando LINQ, un componente de la plataforma .Net que permite agregar consultas nativas semejantes a las de SQL a lenguajes de .Net Framework como es C#. Esta consulta al objeto contexto devuelve un objeto DTO a cuyos campos podemos acceder desde la aplicación.

4.2.2.2 MovilidadPresentacion

Esta capa representa la lógica e interfaz de la aplicación.

Contiene todos los formularios Windows Forms de la aplicación que llevan una funcionalidad asociada (cuando se muestra el formulario, cuando el usuario interactúa con el formulario, etc).

Es en estos formularios en los que se hace uso de los controles proporcionados por Infragistics, como por ejemplo la rejilla de datos UltraGrid, los botones UltraButton, etc. y de las clases de mapeo de datos mencionadas en el punto anterior.

Además, se utilizan las librerías para .Net de MapPoint para la inclusión de mapas en los formularios.

4.2.2.3 Diagrama UML

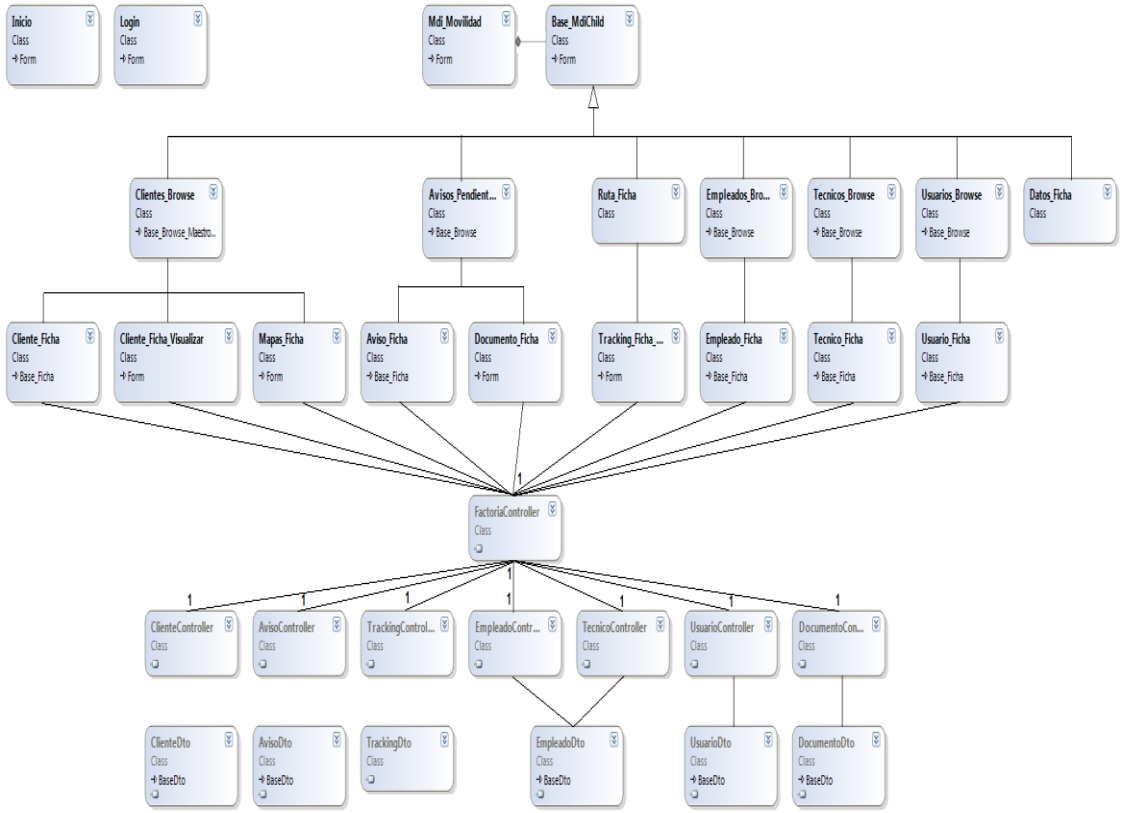


Figura 78. MovilidadPresentacion: Diagrama UML.

El diagrama de la aplicación abarca varias clases y se explicará brevemente a continuación:

En orden de aparición se presentan los formularios de inicio y login. A continuación se muestra el formulario Mdi_Movilidad:

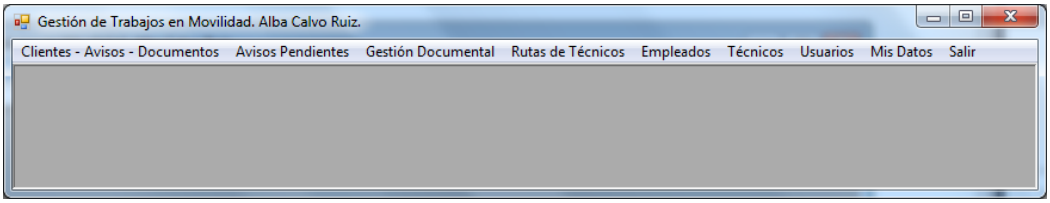


Figura 79. Aplicación cliente .Net: Formulario MDI 1.

Este formulario tiene comportamiento MDI, lo que quiere decir que sus ventanas hijas (MdiChild) residen bajo una única ventana madre. Con esto se consigue que al pulsar en cada una de las pestañas del menú puedan mantenerse a la vez abiertas varias ventanas hijas en el mismo espacio como se muestra en la imagen inferior:

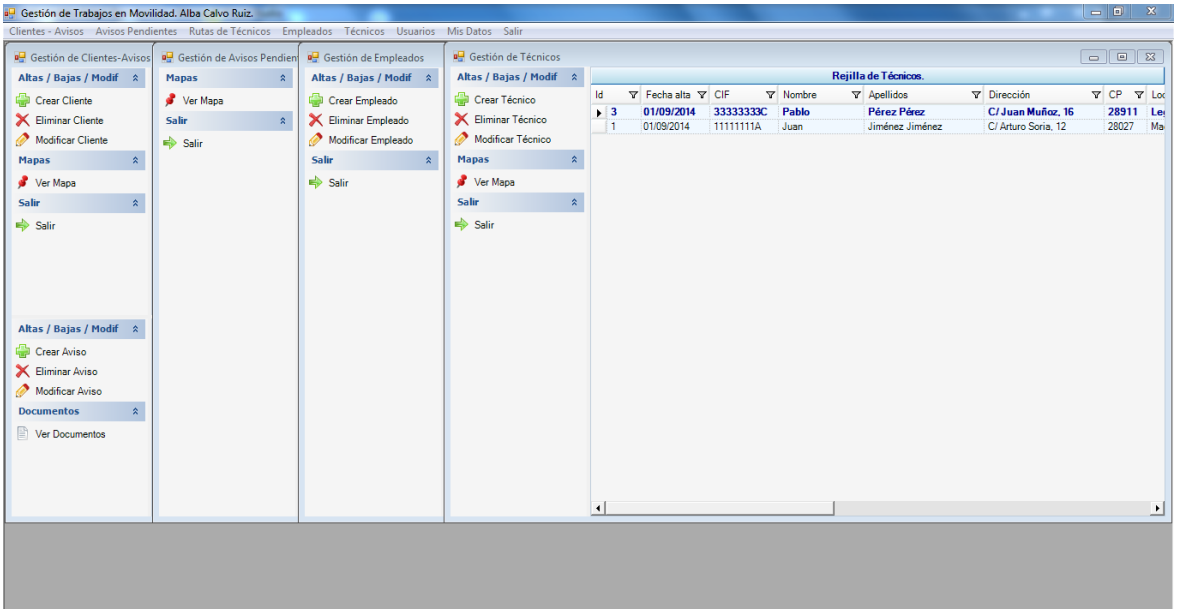


Figura 80. Aplicación cliente .Net: Formulario MDI 2.

Este es el motivo de que todas las clases acabadas en _Browse que se encargan de presentar el listado de clientes-avisos, empleados… sean del tipo Base_MdiChild como muestra la relación is_a del diagrama UML. (Lo mismo ocurre con algunas fichas no asociadas a acciones de rejilla como la de datos personales o la de elección de parámetros de ruta de técnicos).

Podemos observar que cada _Browse tiene una clase _Ficha asociada. Esta es la ficha de alta, modificación y baja que se presenta al hacer click en las opciones de la barra lateral izquierda de cada MdiChild y que tendrá un aspecto y una habilitación de campos distinta según la opción escogida. Además, algunos _Browse están asociados a otro tipo de fichas.

Un ejemplo es Clientes_Browse, que tiene asociada la ficha Mapas_Ficha para la representación cartográfica. Avisos_Browse consta de la ficha Documento_Ficha para la presentación de documentos asociados a avisos.

Cada una de estas fichas está relacionada con un objeto de transmisión de datos DTO que ejerce de intermediario entre la aplicación y la base de datos. Así, el formulario accederá al DTO correspondiente a la ficha (ej: ClienteDTO si nos encontramos en Cliente_Ficha) y se encargará de gestionar la correspondencia entre los campos del DTO y los de la ficha y de realizar las modificaciones pertinentes en la base de datos.

El acceso a los DTO se realizará a través de la clase FactoriaController, una factoría que evita la replicación de los mismos. Esta factoría se encarga de la creación de clases XController sólo si no han sido instanciadas previamente o de la devolución de la instancia existente en caso contrario.

4.2.2.4 Diagrama de secuencia

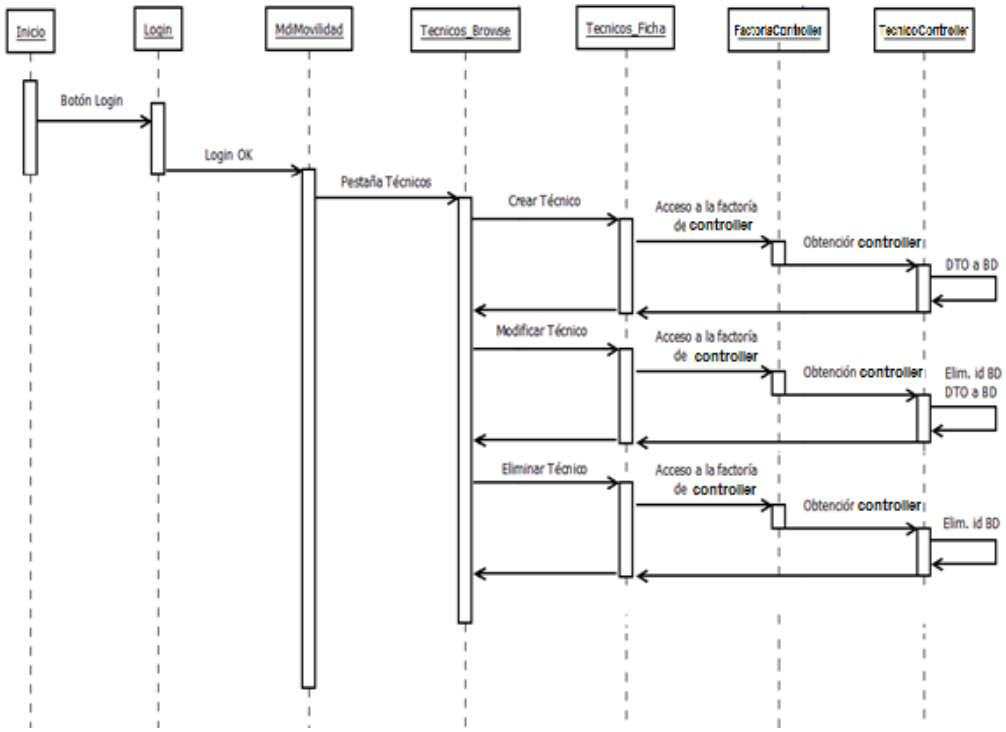


Figura 81. Aplicación cliente .Net: Diagrama de secuencia.

Este es el diagrama de secuencia de una de las posibles opciones: la autenticación y la gestión de técnicos. Es muy representativo y el resto de opciones presenta un flujo muy similar, por lo que no se incluye un diagrama para cada opción.

4.3 API Web

4.3.1 Tecnologías empleadas

Para la realización de la API de servicios Web se ha elegido la tecnología REST debido a su soporte para la utilización de ficheros JSON (Javascript Object Notation) como almacén de información, mucho más ligeros que los ficheros XML.

REST es independiente del lenguaje de programación. He optado por la utilización de PHP debido a su curva de aprendizaje rápida y a que ya contaba con una implementación en PHP de funciones de acceso a la base de datos SQL Server de la versión 1.0 de GTM realizada en mi Trabajo de Fin de Grado.

Para facilitar la tarea de realización de la API, he utilizado el framework Slim, que simplifica el mapeo de peticiones HTTP GET y POST a operaciones sobre la base de datos

4.3.2 Estructura

La estructura de la API Web se resume en dos clases:

- Conexión.class.php: clase para la conexión y relización de consultas a la base de datos.
- restApi.php: clase de mapeo de peticiones HTTP a operaciones sobre la base de datos.

4.3.2.1 Conexion.class.php

Esta clase proporciona una serie de métodos de interactuación con la base de datos: conectar, ejecutar, obtener_resultado, obtener_fila, getCountResult.

```
<?php
    /* Clase encargada de gestionar las conexiones a la base de datos */
    Class Conexion {
        private $dsn = "Driver={SQL Server};Server=[redacted];Database=
                        [redacted];Integrated
                        Security=SSPI;Persist Security Info=False;";
        private $conexion;
        private $result;
        private $array;
        static $_instance;

        /* Función para evitar que el objeto pueda ser creado mediante new*/
        private function __construct(){
            $this->conectar();
        }


        /* Patrón Singleton. Evita la clonación del objeto */
        private function __clone(){ }

        /*Función encargada de crear, si es necesario, el objeto */
        public static function getInstance(){
           if (!(self::$_instance instanceof self))
              self::$_instance=new self();
```

```
            return self::$_instance;
        }

        /* Realiza la conexión a la base de datos */
        private function conectar(){
            $this->conexion = odbc_connect($this->dsn, [redacted], [redacted]);
            if (!$this->conexion)
                exit("<strong>Error al intentar conectar con el origen de datos
                     ODBC.</strong>");
        }

        /* Método para ejecutar una sentencia SQL */
        public function ejecutar($sql){
            $this->result = odbc_exec($this->conexion,$sql)or die(exit("Error en
                            odbc_exec"));
            return $this->result;
        }

       /* Método para obtener el resultado de la sentencia SQL */
       public function obtener_resultado($result){
            $this->array=odbc_result($result, 1);
            return $this->array;
       }

       /* Método para obtener una fila de resultados de la sentencia SQL */
       public function obtener_fila($result,$fila){
           $this->array=odbc_fetch_array($result,$fila);
           return $this->array;
       }

       /* Método para obtener el número de filas de la sentencia SQL ejecutada */
       public function getCountResult() {
            return odbc_num_rows($this->result);
       }
    }
?>
```

Para utilizar esta clase en los ficheros PHP es necesario incluir la sentencia “require 'Conexion.class.php';” e instanciar una variable de conexión:
$conexion = Conexion::getInstance();

Tras realizar esto ya se puede llamar a los métodos de la clase para ejecutar consultas ($conexion->ejecutar(“SELECT * FROM usuario;”);).

4.3.2.2 restApi.php

La API REST se basa en el mapeo de peticiones HTTP a la realización de acciones sobre la base de datos. Las opciones más comunes son:

GET	Obtención de un recurso
POST	Creación de un nuevo recurso
PUT	Actualización de un recurso existente
DELETE	Eliminación de un recurso

El servidor Web donde he alojado la base de datos y la API sólo soporta peticiones GET y POST, por lo que he utilizado el método POST para la creación, actualización y eliminación de recursos.

En REST, las URLs de cada recurso deben estar bien formadas y deben ser únicas.

Las URLs de la API son las siguientes:

GET http://www.albaproyectos.somee.com/SLIM_REST_API/restApi.php/clientes Obtiene todos los clientes. Devuelve un array JSON.
GET http://www.albaproyectos.somee.com/SLIM_REST_API/restApi.php/avisos Obtención de todos los avisos. Devuelve un array JSON.
GET http://www.albaproyectos.somee.com/SLIM_REST_API/restApi.php/avisos/tecnico/:idTecnico Obtención de todos los avisos de un técnico con id = idTecnico. Devuelve un array JSON.
GET http://www.albaproyectos.somee.com/SLIM_REST_API/restApi.php/avisos/tecnico/:idTecnico/:situacion Obtención de todos los avisos de un técnico con id = idTecnico con una situación = situacion (PENDIENTE, REALIZADO, ANULADO). Devuelve un array JSON.
GET http://www.albaproyectos.somee.com/SLIM_REST_API/restApi.php/avisos/tecnico/:idTecnico/:situacion/:leidoSN' Obtención de los avisos de un técnico con id = idTecnico, situación = situacion y valor binario de lectura leidoSN = 0 o 1. Devuelve un array JSON.
POST http://www.albaproyectos.somee.com/SLIM_REST_API/restApi.php/avisos/:idAviso Modificación de un aviso con id = idAviso. Recibe un aviso en formato JSON.

GET
http://www.albaproyectos.somee.com/SLIM_REST_API/restApi.php/documentos/getDocumento/:idAviso/:rutaRelativa/:tipoDocumento

Obtiene un documento asociado con el aviso con id = idAviso, ruta = rutaRelativa y tipo = tipoDocumento (FOTO o TICKET). Devuelve el número de filas en formato JSON.

POST
http://www.albaproyectos.somee.com/SLIM_REST_API/restApi.php/documentos/subirDocumento

Inserción de un documento. Recibe un documento en formato JSON.

POST http://www.albaproyectos.somee.com/SLIM_REST_API/restApi.php/loginTecnico

Comprobación de credenciales de usuario. Recibe un objeto JSON con los datos del usuario y devuelve el id autenticado o -1 si las credenciales son incorrectas codificado en JSON.

POST
http://www.albaproyectos.somee.com/SLIM_REST_API/restApi.php/tracking/:idTecnico

Inserción de coordenadas geográficas asociadas a un técnico con id = idTecnico. Recibe un objeto JSON con los datos de geolocalización. Devuelve el número de filas insertadas.

Con el framework Slim, realizar el mapeo de las URLs citadas es tan sencillo como instanciar un objeto Slim y ejecutar sus métodos get y post para asociar una URL a una función en PHP. Por ejemplo:

```
<?php

    require 'Slim/Slim.php';
    $app = new \Slim\Slim();

    $app->get('/avisos', 'getAvisos');

    …

    //GET http://www.albaproyectos.somee.com/SLIM_REST_API/restApi.php/avisos
    function getAvisos() {
       $sql = "SELECT * FROM aviso;";
       jsonFromSqlQuery($sql);
    }

    …

    $app->run();

?>
```

4.4 Aplicación de geolocalización y gestión de avisos en terminales en movilidad

4.4.1 Tecnologías empleadas

La aplicación móvil se ha desarrollado en Android debido a su posición dominante en el mercado de los dispositivos móviles (líder en prestaciones, alta implantación, sistema operativo de libre distribución, lenguaje de desarrollo Java).

Para ello se ha trabajado con el entorno Eclipse para desarrollo de aplicaciones Android.

4.4.2 Estructura

La aplicación Android consta de varias partes fundamentales:

- Clases para la sincronización de la base de datos y el repositorio de datos locales con la base de datos y el repositorio de ficheros alojados en el servidor Web.
- Tarea asíncrona de login.
- Servicio de tracking.
- Actividades y layouts que definen la interfaz de usuario.

Además, cuenta con utilidades para la comunicación con la API Web utilizando JSON como contenedor de datos y con la definición de controles Android avanzados, como el cuadro editable de firma.

Las partes más importantes se explican a continuación.

4.4.3 Sincronización de base de datos y repositorio de ficheros locales con los alojados en el servidor Web

Para permitir el trabajo offline de los técnicos y la sincronización de datos entre sus terminales en movilidad y el servidor Web se ha utilizado el patrón SyncAdapter[24], que se ilustra en el siguiente esquema:

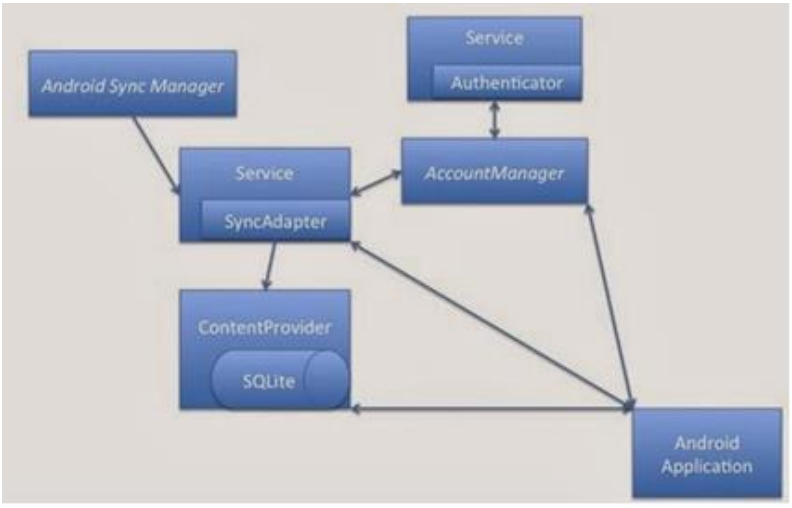


Figura 82. Aplicación cliente Android: Sincronización. Patrón SyncAdapter.

[24] http://developer.android.com/training/sync-adapters/creating-sync-adapter.html

Básicamente, el funcionamiento del patrón se basa en la asociación de una cuenta a la aplicación Android para la realización de la sincronización.

Las cuentas de sincronización del teléfono son gestionadas de forma automática por el gestor de cuentas AccountManager del teléfono.

El dispositivo dispone además de un gestor de sincronización SyncManager que se encarga de invocar al adaptador de sincronización SyncAdapter asociado a cada cuenta.

El funcionamiento de este adaptador está definido por el programador en el código. En esencia, el adaptador define una serie de acciones de sincronización entre la base de datos SQLite y el repositorio de ficheros locales del dispositivo, y la base de datos y el repositorio de ficheros alojados en el servidor Web. Para ello hace uso de la API Web y de un proveedor de contenidos ContentProvider que define la estructura y las acciones de lectura y escritura sobre la base de datos SQLite del teléfono.

Así, en el código sería necesario implementar tres clases:

- Un servicio que gestione la información de la cuenta asociada a la aplicación Android, proporcionando un identificador único que utilizará el adaptador de sincronización SyncAdapter para operar.

- Un adaptador de sincronización SyncAdapter a la escucha de sincronizaciones periódicas o forzadas por el usuario desde la interfaz de usuario.

- Un proveedor de contenido que defina la estructura de la base de datos SQLite y las acciones de inserción, actualización y modificación de los registros de cada una de las tablas de la misma.

4.4.4 Tarea asíncrona de login

Al iniciar la aplicación, se muestra al usuario la ventana asociada a la actividad Login, donde el técnico introduce sus credenciales (nick y password).

A continuación, se comprueba que los datos son correctos. Esta comprobación se realiza mediante la llamada a una tarea asíncrona que se ejecuta en otro hilo, evitando que la interfaz de la aplicación no se vea ralentizada dando impresión de mal funcionamiento.

Esta tarea muestra un diálogo de proceso mientras comprueba las credenciales del usuario. Para ello se comunica con la API Web, enviándole un objeto JSON con los datos del usuario por método POST y recibiendo otro objeto JSON con un código de respuesta (id del usuario autenticado si el login es correcto o -1 si los datos no son correctos).

Si el login es incorrecto, se mantiene al usuario en la ventana de login mostrando un mensaje de error. En cambio, si la autenticación es correcta, se guarda el id de usuario en el fichero de preferencias de Android y se dirige al usuario al menú principal de la aplicación.

Almacenamiento del id de usuario en el fichero de preferencias de Android:

```
SharedPreferences settings = PreferenceManager.getDefaultSharedPreferences(contexto);
SharedPreferences.Editor editor = settings.edit();
editor.putString("id_usu", aux);
editor.commit();
```

El almacenamiento del id de usuario en el fichero de preferencias permite que la aplicación memorice el usuario autenticado y que éste no necesite hacer login cada vez que ejecute la aplicación en su terminal. La única forma de borrar las preferencias es saliendo de la aplicación con la opción de logout desde el menú principal.

4.4.5 Servicio de tracking

La aplicación cuenta con un menú donde el técnico tiene la opción de iniciar y finalizar la geolocalización de su dispositivo. Este menú se encarga de gestionar la vida un servicio (TrackingService) que envía las coordenadas GPS del técnico de forma periódica cada minuto si ha recorrido una distancia de al menos cien metros.

Este servicio se ejecuta en un hilo diferente al de la aplicación principal, por lo que si ésta pasa a segundo plano o es finalizada por el planificador de procesos de Android, el servicio seguirá activo.

Para utilizar este servicio basta con declarar una variable de tipo Intent para la clase TrackingService y llamar al método startService para iniciar el servicio o stopService para finalizarlo:

```
Intent intent = new Intent(this, TrackingService.class);
startService(intent);
stopService(intent);
```

La clase ha debe extender Service y sobrescribir ciertos métodos que todo servicio debe tener (onBind, onStartCommand).

A grandes rasgos, el funcionamiento del servicio es el siguiente:

Primero se inicializa una variable de tipo LocationManager que da acceso a los servicios de localización del sistema.

```
LocationManager locManager;
locManager = (LocationManager) getSystemService (Context.LOCATION_SERVICE);
```

Los servicios de localización del sistema permiten obtener actualizaciones periódicas de la posición geográfica del dispositivo. Esto se hace mediante la creación de un "escuchador" LocationListener que obtiene la posición mediante el GPS del dispositivo mediante el método requestLocationUpdates, donde LOCATION_INTERVAL es el intervalo de petición de actualizaciones (un minuto) y LOCATION_DISTANCE es la distancia de petición de actualizaciones (cien metros):

```
LocationListener locListener=new LocationListener(LocationManager.GPS_PROVIDER);
locManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,LOCATION_INTERVAL,
                                  LOCATION_DISTANCE, locListener);
```

Este LocationListener implementa la clase android.location.LocationListener y sobreescribe algunos de sus métodos, entre ellos onLocationChanged, que ante un cambio de posición del dispositivo, se encarga de almacenar las coordenadas geográficas en la base de datos Web. Para ello, obtiene la longitud y la latitud de la posición recibida y la compara con la última calculada (-1, -1 inicialmente). Si la posición no dista más de cien metros de la anterior, se incrementa un contador de repeticiones que simbolizan los minutos que permanece el técnico en una misma localización. De lo contrario, se realiza la inserción de la posición calculada así como de la posición anterior que no se había insertado aún en la base de datos. Esta inserción se realiza mediante el uso de la API Web, enviando un objeto JSON con los datos de las coordenadas geográficas.

4.4.6 Interfaz de usuario

La aplicación cuenta con varias clases Java y con varios layouts XML que configuran las ventanas que se muestran al usuario y que hacen uso de los servicios de login, sincronización y tracking mencionados en los puntos anteriores.

4.4.7 Diagramas

4.4.7.1 Diagrama UML

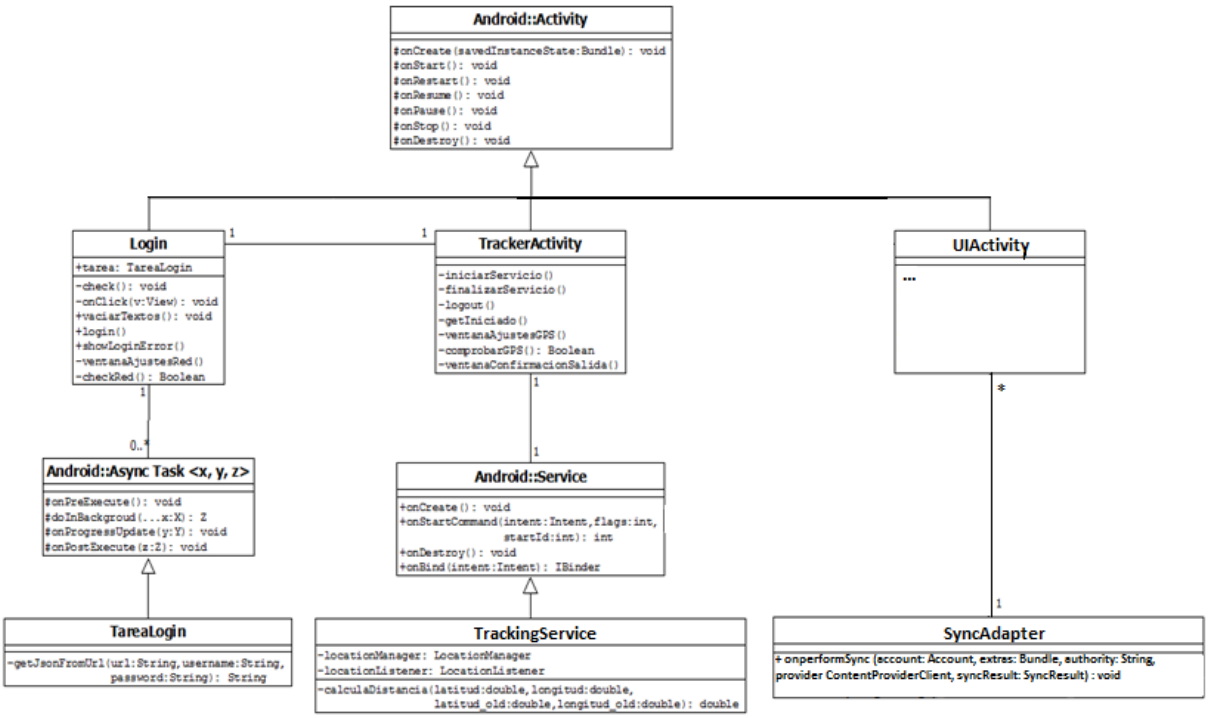


Figura 83. Aplicación cliente Android: Diagrama UML.

4.4.7.2 Diagramas de secuencia

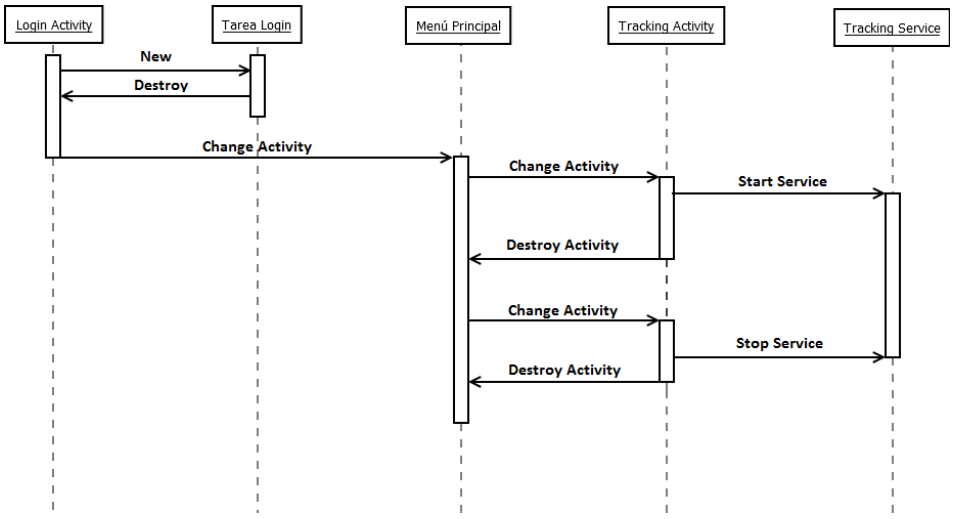


Figura 84. Aplicación cliente Android: Diagrama de secuencia de tracking.

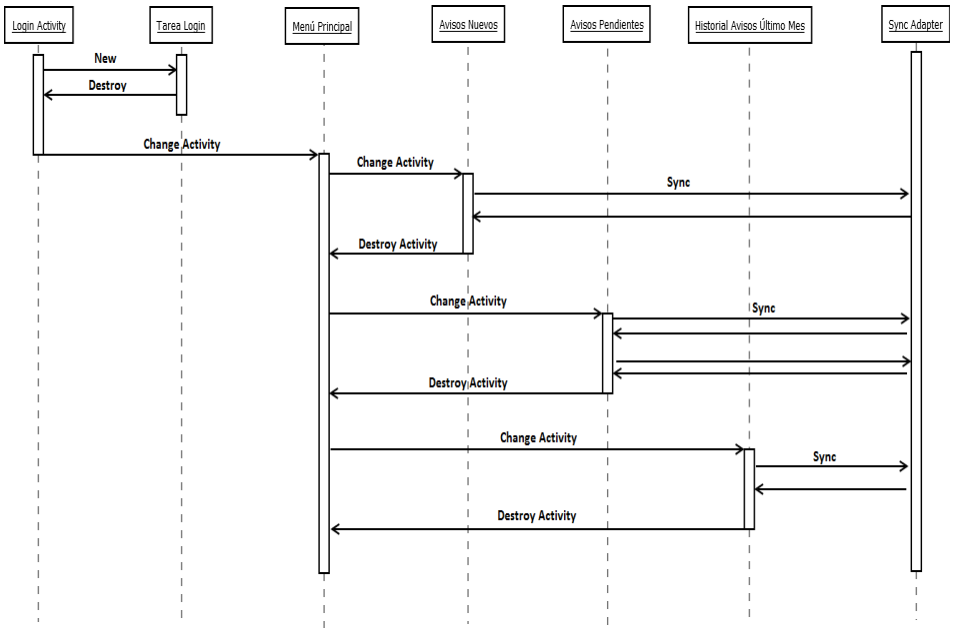


Figura 85. Aplicación cliente Android: Diagrama de secuencia de gestión de avisos.

5 Integración, pruebas y resultados

Para probar el correcto funcionamiento de la aplicación se han realizado pruebas individuales a cada componente del proyecto (aplicación cliente C#, API Web y aplicación móvil de tracking) y pruebas de integración de conjunto.

Para realizar estas pruebas ha sido necesario contratar un hosting para simular el comportamiento de la aplicación a pequeña escala. Este hosting incluye el soporte para poder alojar una base de datos SQLServer de pruebas de hasta 3 MB y el dominio "albaproyectos.somee.com" donde poder alojar la página Web. Además se ha contratado un buzón de correo que permite dar de alta hasta 10 usuarios.

5.1 Pruebas modulares

5.1.1 Pruebas de aplicación cliente de gestión

5.1.1.1 Entorno de prueba

Para realizar esta prueba se necesita un entorno con las siguientes características:

- Equipos con sistemas operativo Windows, preferiblemente la versión 7.
- SQLServer 2012 y Management Studio
- .Net Framework 4 o superior.
- Visual Studio 2010
- Infragistics versión 2011.2

5.1.1.2 Pruebas realizadas

5.1.1.2.1 Pruebas de correcto almacenamiento en la base de datos

5.1.1.2.1.1. Diseño de la prueba

La prueba consiste en la consulta de la base de datos mediante sentencias SQL utilizando la herramienta SQL Server Management Studio para comprobar el alta, baja o modificación de registros de la base de datos.

Un ejemplo de prueba sería la utilización del formulario de gestión de empleados por parte del administrador y la comprobación de la correcta realización de las acciones de alta, baja y modificación, comprobando su repercusión en la base de datos.

5.1.1.2.1.2. Análisis de los resultados

Al iniciar la prueba el formulario y la tabla de empleados se encuentran de la siguiente forma:

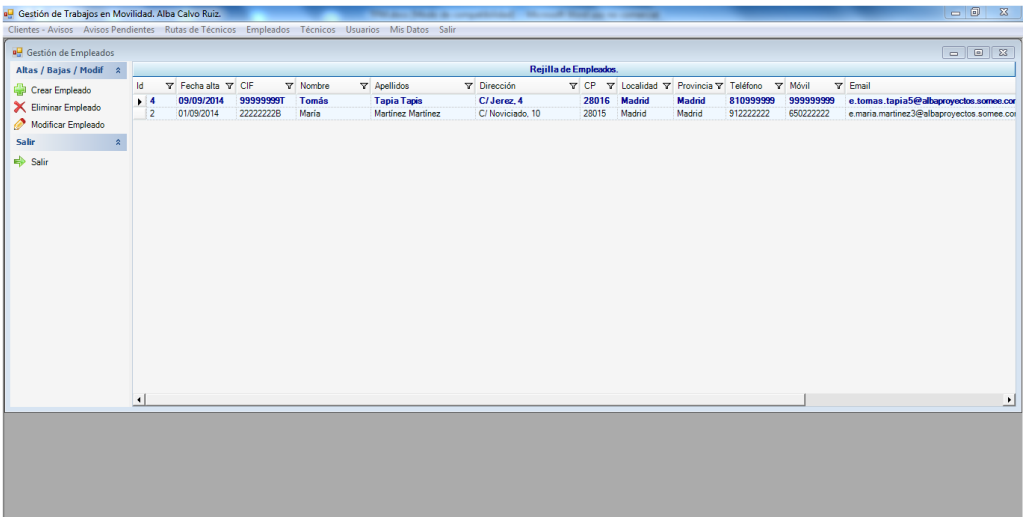


Figura 86. Pruebas modulares. Prueba de aplicación cliente .Net: Rejilla de empleados.

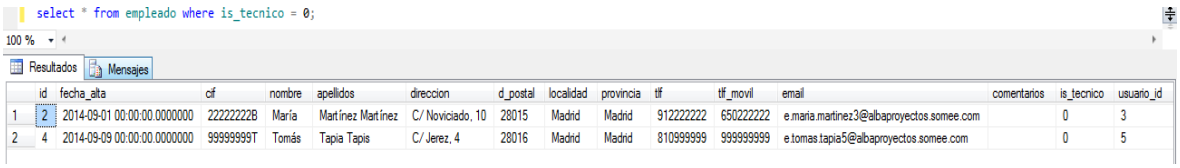


Figura 87. Pruebas modulares. Prueba de aplicación cliente .Net: Consulta de empleados.

La tabla empleados está estrechamente relacionada con la tabla usuario, pues la creación de un empleado supone la creación de su usuario asociado. Esta tabla se encuentra inicialmente de la siguiente forma:

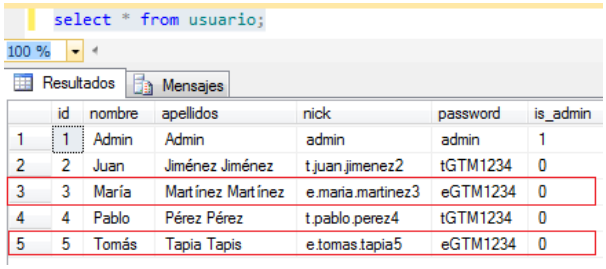


Figura 88. Pruebas modulares. Prueba de aplicación cliente .Net: Consulta de usuarios.

A continuación se analizará el resultado de las acciones de la barra lateral izquierda del formulario:

- Crear empleado:

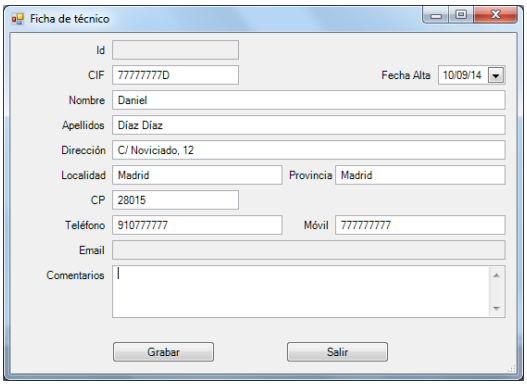
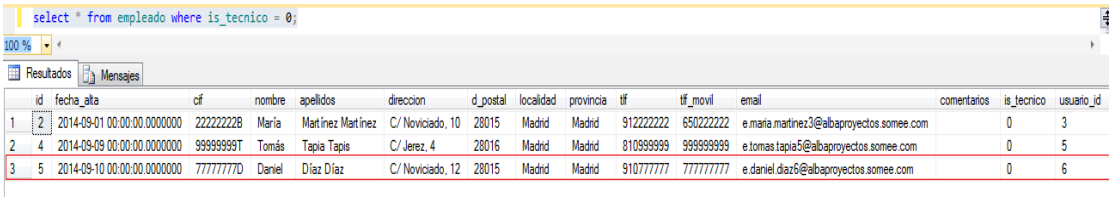


Figura 89. Pruebas modulares. Prueba de aplicación cliente .Net: Alta de empleado.



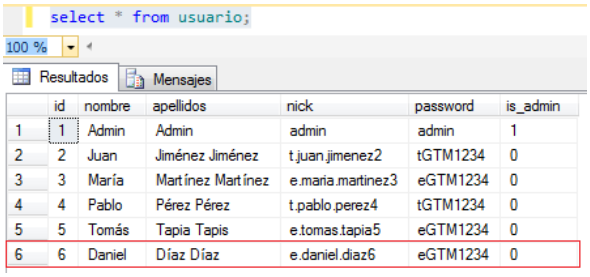
```
select * from empleado where is_tecnico = 0;
```

100 %

Resultados | Mensajes

	id	fecha_alta	cif	nombre	apellidos	direccion	d_postal	localidad	provincia	tlf	tlf_movil	email	comentarios	is_tecnico	usuario_id
1	2	2014-09-01 00:00:00.0000000	22222222B	María	Martínez Martínez	C/ Noviciado, 10	28015	Madrid	Madrid	912222222	650222222	e.maria.martinez3@albaproyectos.somee.com		0	3
2	4	2014-09-09 00:00:00.0000000	99999999T	Tomás	Tapia Tapis	C/ Jerez, 4	28016	Madrid	Madrid	810999999	999999999	e.tomas.tapia5@albaproyectos.somee.com		0	5
3	5	2014-09-10 00:00:00.0000000	77777777D	Daniel	Díaz Díaz	C/ Noviciado, 12	28015	Madrid	Madrid	910777777	777777777	e.daniel.diaz6@albaproyectos.somee.com		0	6

Figura 90. Pruebas modulares. Prueba de aplicación cliente .Net: Consulta de empleados con cambios reflejados.



```
select * from usuario;
```

100 %

Resultados | Mensajes

	id	nombre	apellidos	nick	password	is_admin
1	1	Admin	Admin	admin	admin	1
2	2	Juan	Jiménez Jiménez	t.juan.jimenez2	tGTM1234	0
3	3	María	Martínez Martínez	e.maria.martinez3	eGTM1234	0
4	4	Pablo	Pérez Pérez	t.pablo.perez4	tGTM1234	0
5	5	Tomás	Tapia Tapis	e.tomas.tapia5	eGTM1234	0
6	6	Daniel	Díaz Díaz	e.daniel.diaz6	eGTM1234	0

Figura 91. Pruebas modulares. Prueba de aplicación cliente: Consulta de usuarios con cambios reflejados.

En las imágenes superiores se puede comprobar que los cambios han surtido efecto en la base de datos y que tanto el empleado como el usuario asociado se han insertado con éxito. Además, ambos formularios se actualizan consecuentemente:

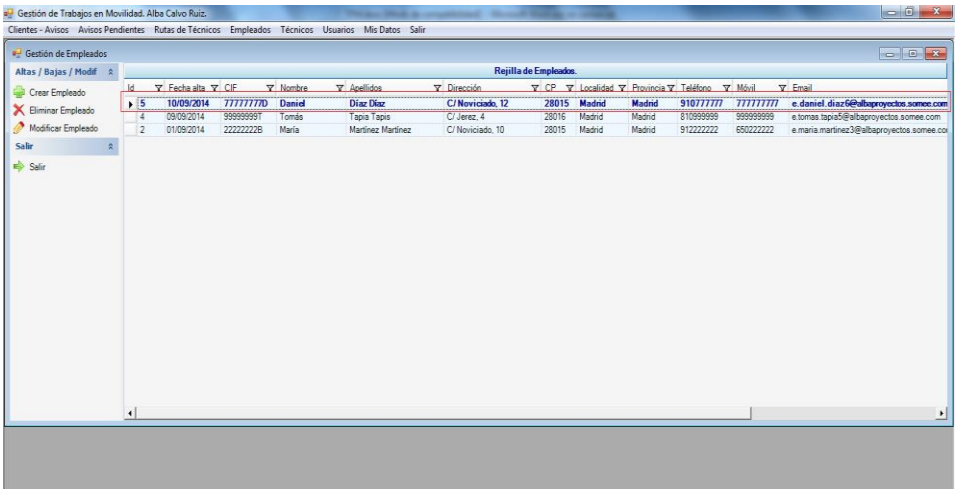


Figura 92. Pruebas modulares. Prueba de aplicación cliente .Net: Rejilla de empleados con cambios reflejados.

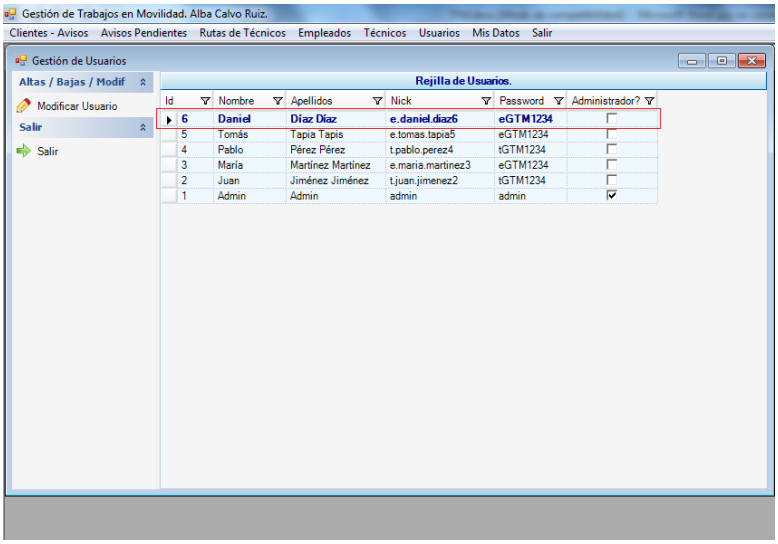


Figura 93. Pruebas modulares. Prueba de aplicación cliente .Net: Rejilla de usuarios con cambios reflejados.

- Modificar empleado

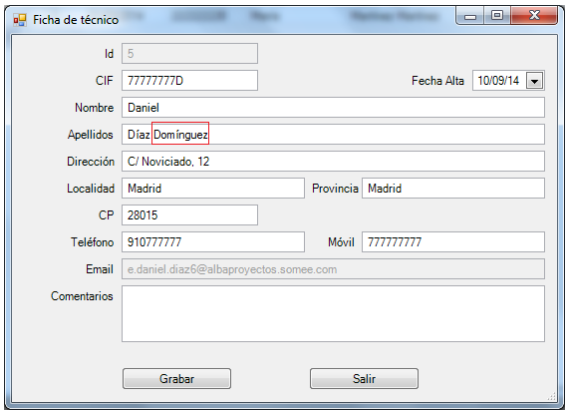
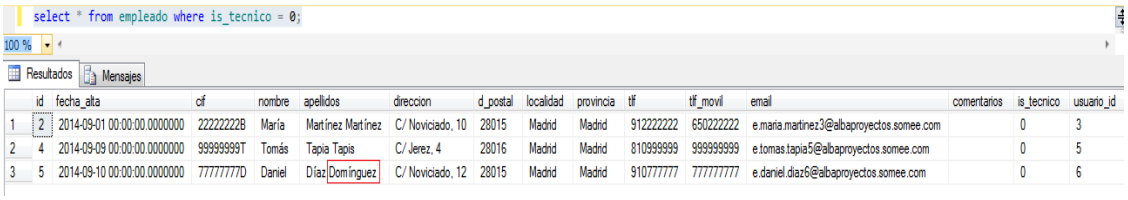


Figura 94. Pruebas modulares. Prueba de aplicación cliente .Net: Modificación de empleado.

Los cambios se realizan de forma exitosa tanto en la base de datos como en los formularios asociados:



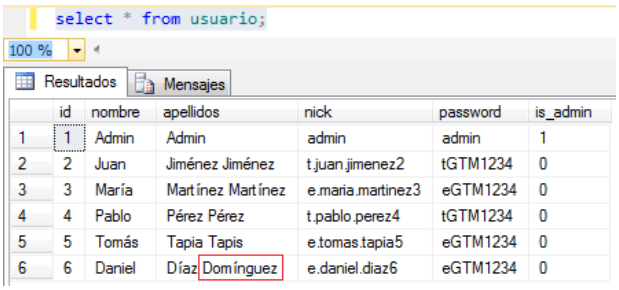
```
select * from empleado where is_tecnico = 0;
```

100 %

Resultados | Mensajes

	id	fecha_alta	cif	nombre	apellidos	direccion	d_postal	localidad	provincia	tlf	tlf_movil	email	comentarios	is_tecnico	usuario_id
1	2	2014-09-01 00:00:00.0000000	22222222B	María	Martínez Martínez	C/ Noviciado, 10	28015	Madrid	Madrid	912222222	650222222	e.maria.martinez3@albaproyectos.somee.com		0	3
2	4	2014-09-09 00:00:00.0000000	99999999T	Tomás	Tapia Tapis	C/ Jerez, 4	28016	Madrid	Madrid	810999999	999999999	e.tomas.tapia5@albaproyectos.somee.com		0	5
3	5	2014-09-10 00:00:00.0000000	77777777D	Daniel	Díaz Domínguez	C/ Noviciado, 12	28015	Madrid	Madrid	910777777	777777777	e.daniel.diaz6@albaproyectos.somee.com		0	6

Figura 95. Pruebas modulares. Prueba de aplicación cliente .Net: Consulta de empleados con cambios reflejados 2.



```
select * from usuario;
```

100 %

Resultados | Mensajes

	id	nombre	apellidos	nick	password	is_admin
1	1	Admin	Admin	admin	admin	1
2	2	Juan	Jiménez Jiménez	t.juan.jimenez2	tGTM1234	0
3	3	María	Martínez Martínez	e.maria.martinez3	eGTM1234	0
4	4	Pablo	Pérez Pérez	t.pablo.perez4	tGTM1234	0
5	5	Tomás	Tapia Tapis	e.tomas.tapia5	eGTM1234	0
6	6	Daniel	Díaz Domínguez	e.daniel.diaz6	eGTM1234	0

Figura 96. Pruebas modulares. Prueba de aplicación cliente .Net: Consulta de usuarios con cambios reflejados 2.

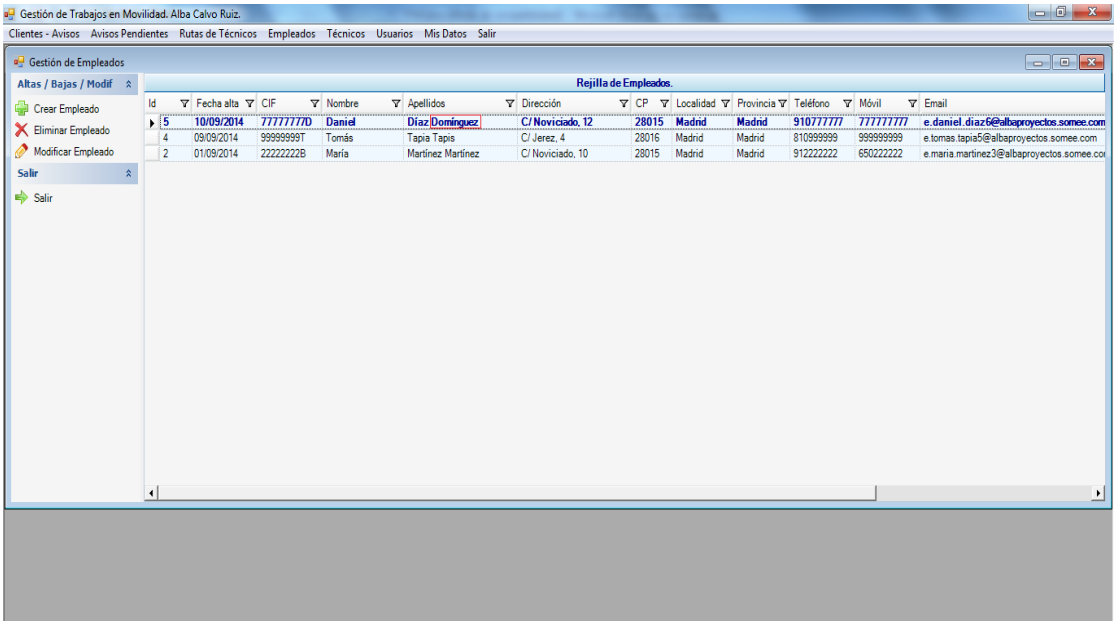


Figura 97. Pruebas modulares. Prueba de aplicación cliente .Net: Rejilla de empleados con cambios reflejados 2.

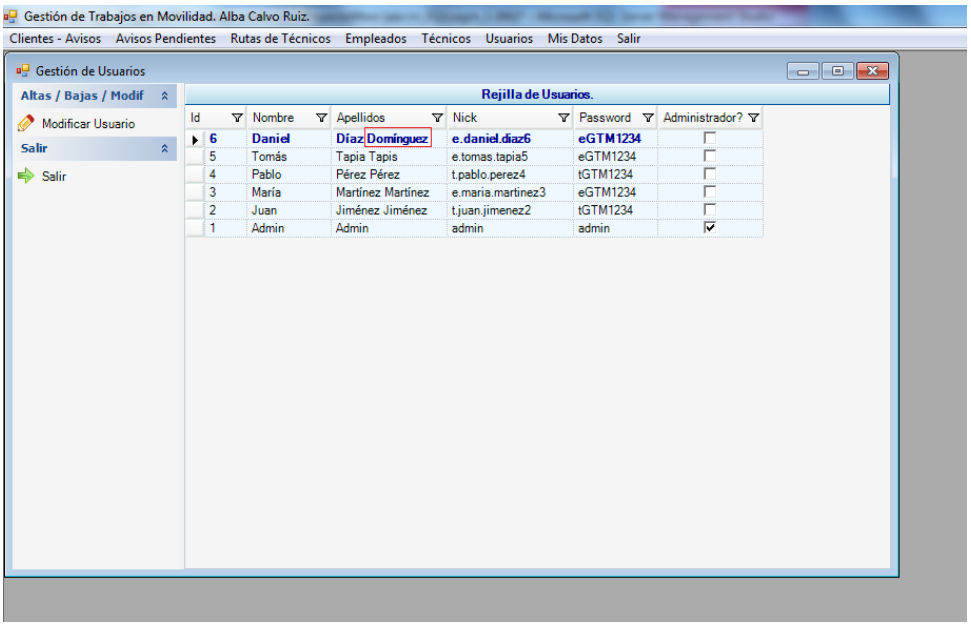


Figura 98. Pruebas modulares. Prueba de aplicación cliente .Net: Rejilla de usuarios con cambios reflejados 2.

- Eliminar empleado

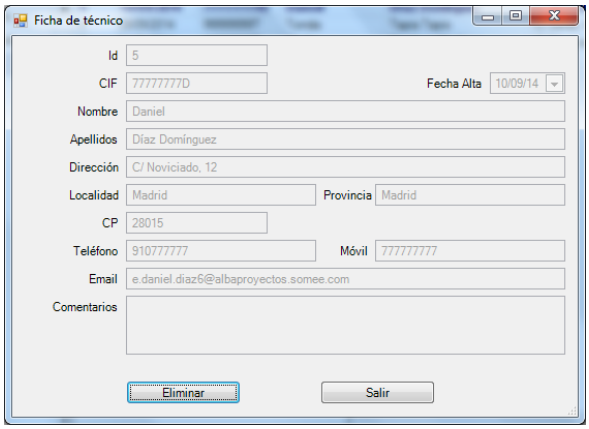


Figura 99. Pruebas modulares. Prueba de aplicación cliente .Net: Eliminación de empleado.

Los cambios se realizan de forma exitosa tanto en la base de datos como en los formularios asociados:

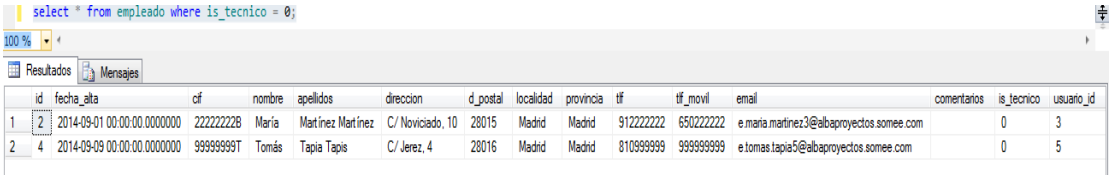


Figura 100. Pruebas modulares. Prueba de aplicación cliente .Net: Consulta de empleados con cambios reflejados 3.

```
select * from usuario;
```

100 %

Resultados | Mensajes

	id	nombre	apellidos	nick	password	is_admin
1	1	Admin	Admin	admin	admin	1
2	2	Juan	Jiménez Jiménez	t.juan.jimenez2	tGTM1234	0
3	3	María	Martínez Martínez	e.maria.martinez3	eGTM1234	0
4	4	Pablo	Pérez Pérez	t.pablo.perez4	tGTM1234	0
5	5	Tomás	Tapia Tapis	e.tomas.tapia5	eGTM1234	0

Figura 101. Pruebas modulares. Prueba de aplicación cliente .Net: Consulta de usuarios con cambios reflejados 3.

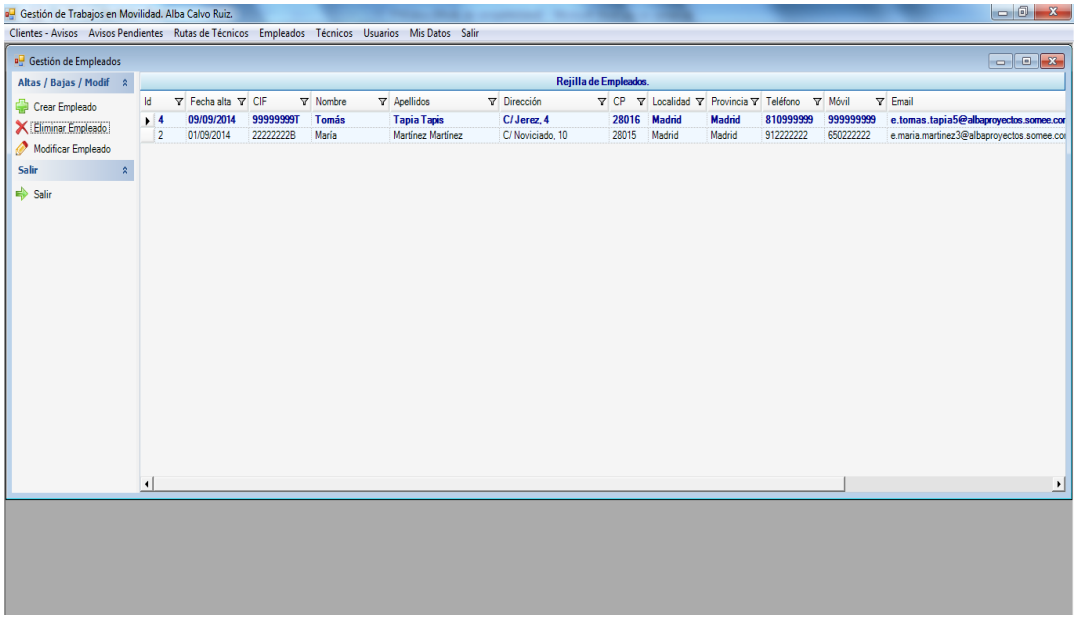


Figura 102. Pruebas modulares. Prueba de aplicación cliente .Net: Rejilla de empleados con cambios reflejados 3.

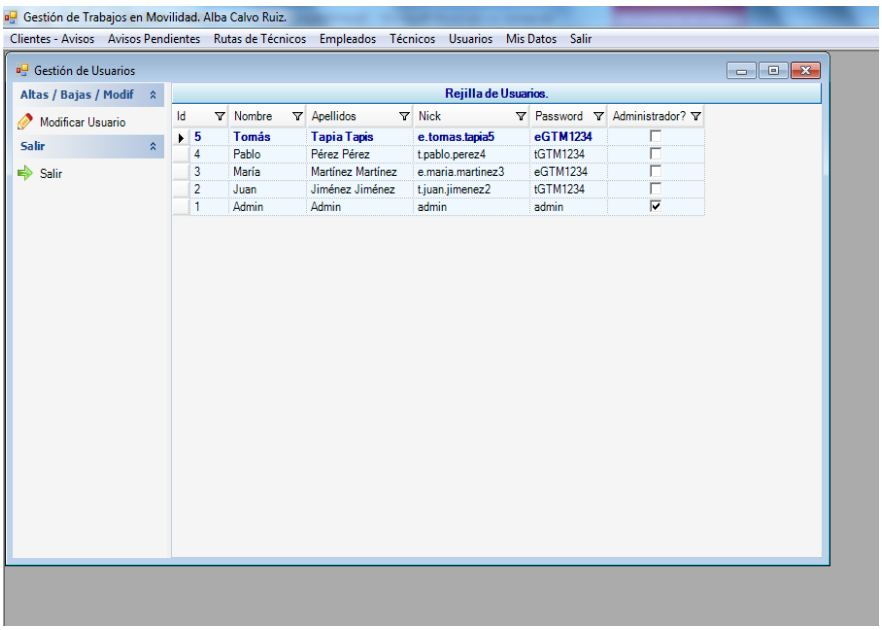


Figura 103. Pruebas modulares. Prueba de aplicación cliente .Net: Rejilla de usuarios con cambios reflejados 3.

5.1.1.2.2 Pruebas de interfaz

5.1.1.2.2.1. Diseño de la prueba

La prueba consiste en la comprobación del correcto funcionamiento de cada uno de los componentes de la interfaz, incluidos los campos de texto con limitación en el número de caracteres.

5.1.1.2.2.2. Análisis de resultados

El correcto funcionamiento de los botones y las rejillas de datos se ha comprobado mediante la ejecución de todos los flujos posibles y el resultado ha sido correcto.

También se ha comprobado en los formularios de alta, baja y modificación la completitud de los campos esenciales y la longitud máxima de los mismos, mostrando un mensaje al usuario en caso de alguna anomalía.

- Campos sin rellenar:

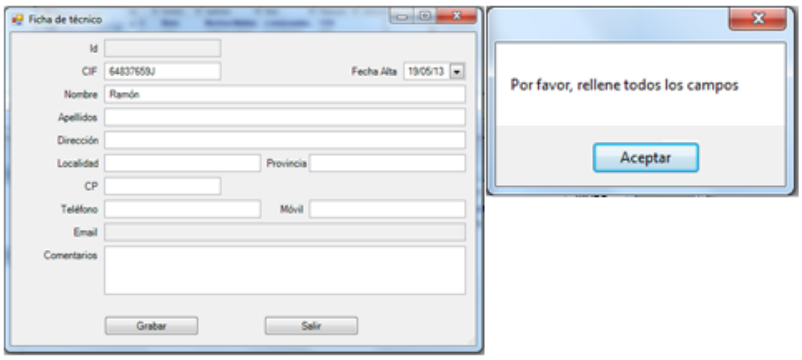


Figura 104. Pruebas modulares. Prueba de aplicación cliente .Net: Prueba de campos sin rellenar.

- Campos demasiado largos:

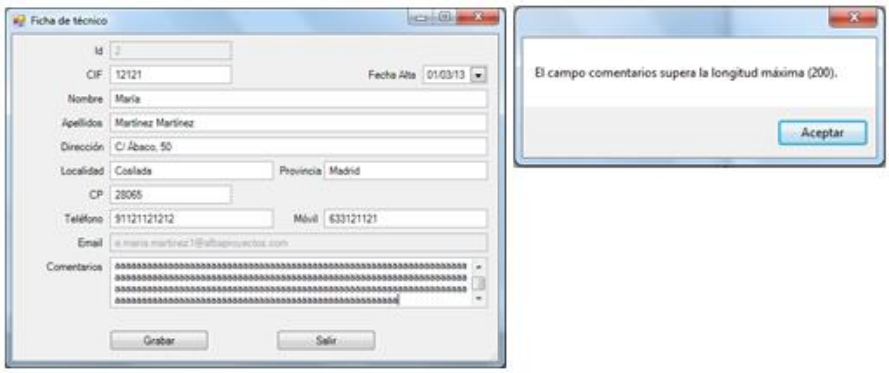


Figura 105. Pruebas modulares. Prueba de aplicación cliente .Net: Prueba de campos demasiado largos.

5.1.1.2.3 Pruebas de presentación de mapas

5.1.1.2.3.1. Diseño de la prueba

La prueba consiste en la comprobación de la correcta resolución de direcciones de clientes en coordenadas geográficas utilizando las librerías de MapPoint, y en la representación de las mismas mediante marcadores en el mapa.

También se ha comprobado la presentación de la ficha asociada a los marcadores del mapa al seleccionarlos.

5.1.1.2.3.2. Análisis de resultados

- Resolución de direcciones de clientes en coordenadas geográficas:

  Al grabar el alta de un cliente, se ha comprobado la correcta obtención y grabación en base de datos de las coordenadas geográficas de su dirección probando direcciones con latitud y longitud conocidas (previa consulta en Google Maps por ejemplo):

```
select * from cliente
```

100 %

Resultados | Mensajes

	id	fecha_alta	cif	nombre	apellidos	direccion	direccion_latitud	direccion_longitud	d_postal	localidad	provinc
1	8	2014-09-03 02:44:18.1314667	55555	Gonzalo	García García	C/ Mar de las Indias, 8	40,4695913940668	-3,86256714351475	28221	Majadahonda	Madrid
2	10	2014-09-10 21:19:35.0364718	88888	Víctor	Valdés Valdés	C/ Hernán Cortés, 5	40,4727708175778	-3,86992536485195	28220	Majadahonda	Madrid
3	12	2014-09-10 21:35:31.9122019	12121	Jorge	Jiménez Jiménez	C/ Isaac Peral, 20	40,475659975782	-3,87391850352287	28220	Majadahonda	Madrid
4	13	2014-09-10 21:38:28.9343270	13131	Francisco	Fuentes Fuentes	C/ San Luis, 18	40,4716882109642	-3,87735323980451	28220	Majadahonda	Madrid
5	14	2014-09-10 23:15:03.2447516	14141	Bárbara	Bastos Bastos	Av Paralela, 23	40,4747037682682	-3,86555193923414	28221	Majadahonda	Madrid
6	15	2014-09-10 23:16:34.4889704	15151	Carlos	Cuevas Cuevas	C/ Las Norias, 24	40,4712408687919	-3,86791563592851	28221	Majadahonda	Madrid
7	16	2014-09-10 23:19:17.0542687	16161	Lorena	Lozano Lozano	C/ Virgen del Rosario, 4	40,4763395804912	-3,88250702060759	28220	Majadahonda	Madrid

Figura 106. Pruebas modulares. Prueba de aplicación cliente .Net: Prueba de presentación de mapas. Consulta de direcciones de clientes.

También se ha probado la correcta ubicación de las mismas en el mapa:

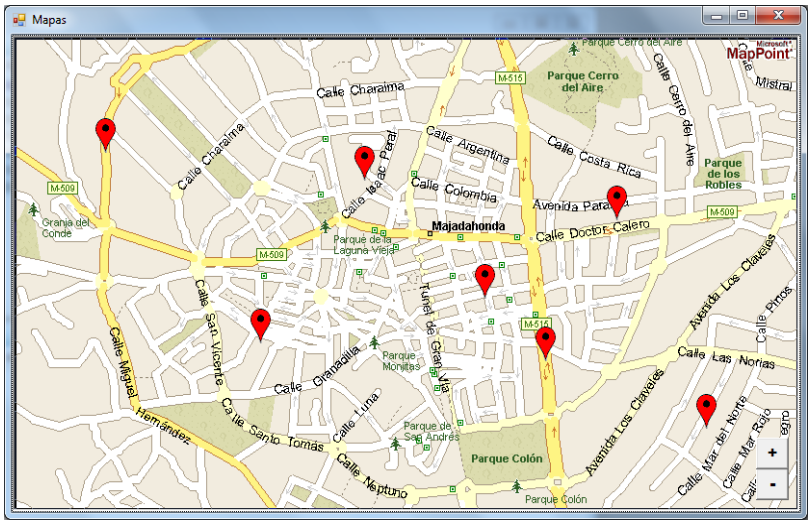


Figura 107. Pruebas modulares. Prueba de aplicación cliente .Net: Prueba de presentación de mapas. Mapa de localización de clientes.

- Presentación de ficha asociada a marcadores.

  Se ha comprobado la presentación de información de cliente, aviso o coordenada según el tipo de marcador en el mapa en una ficha emergente:

  - Ficha de información de cliente:

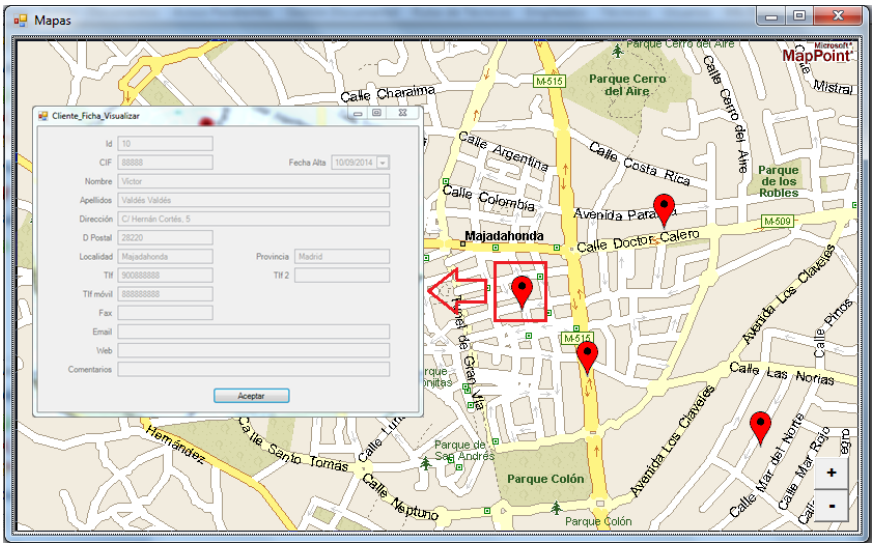


Figura 108. Pruebas modulares. Prueba de aplicación cliente .Net: Prueba de presentación de mapas. Ficha de información de cliente.

  - Ficha de información de aviso:

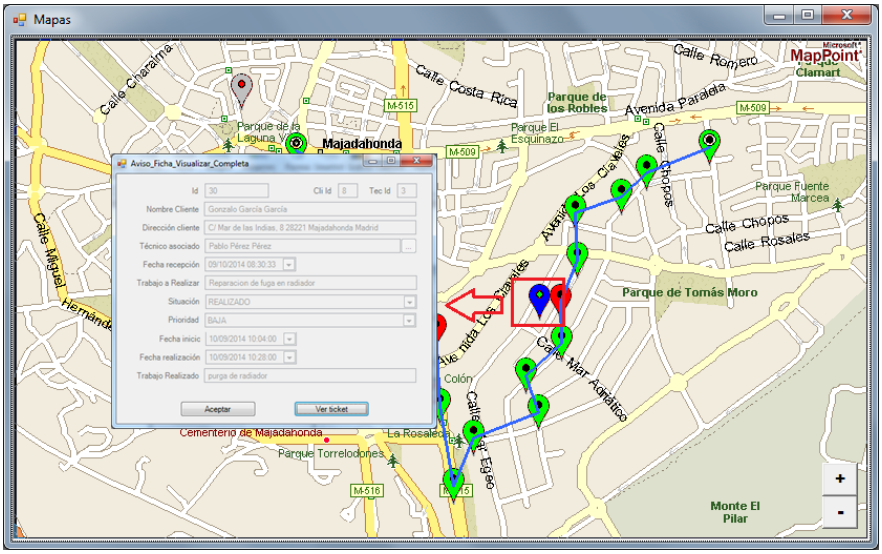


Figura 109. Pruebas modulares. Prueba de aplicación cliente .Net: Prueba de presentación de mapas. Ficha de información de aviso.

- Ficha de información de parada en ruta:

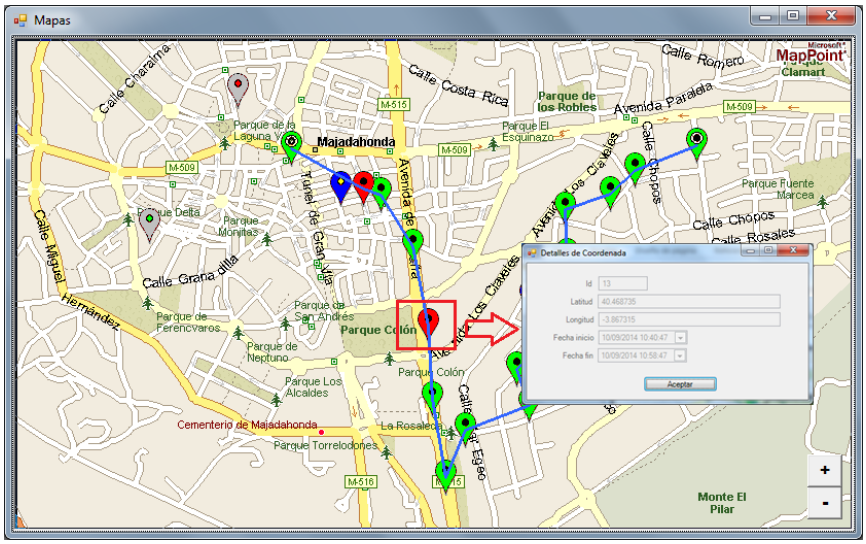


Figura 110. Pruebas modulares. Prueba de aplicación cliente .Net: Prueba de presentación de mapas. Ficha de información de parada.

5.1.1.2.4 Pruebas de visualización de documentos

5.1.1.2.4.1. Diseño de la prueba

La prueba consiste en la comprobación de la correcta visualización de los documentos ubicados en el servidor desde la aplicación cliente.

5.1.1.2.4.2. Análisis de resultados

Se han comprobado desde la aplicación Web para la gestión del dominio somee.com los documentos asociados a cada aviso (el nombre tiene la forma [idAviso]_[fecha]_[tipoDocumento].png) y su correcta consulta desde la aplicación cliente.

Ej. Documento ticket asociado al aviso 1:

- En el repositorio de ficheros Web se observa que hay un ticket asociado al aviso 1:

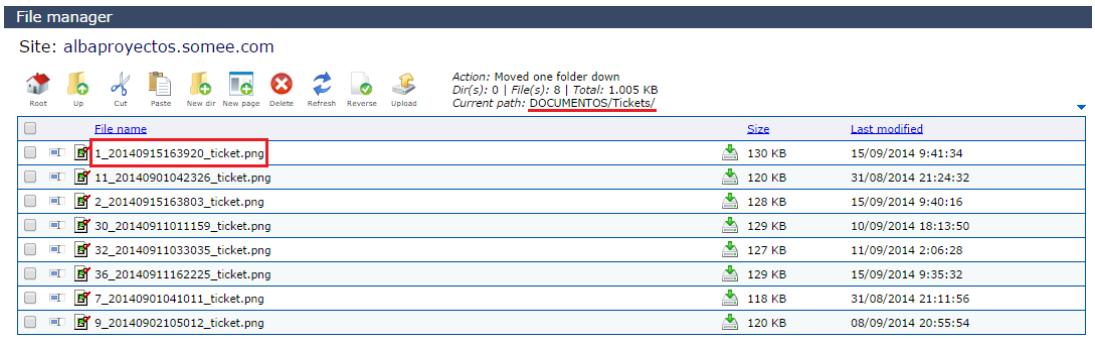


Figura 111. Pruebas modulares. Prueba de aplicación cliente .Net: Prueba de visualización de documentos. Documentos del repositorio Web.

Es posible visualizarlo en el navegador:

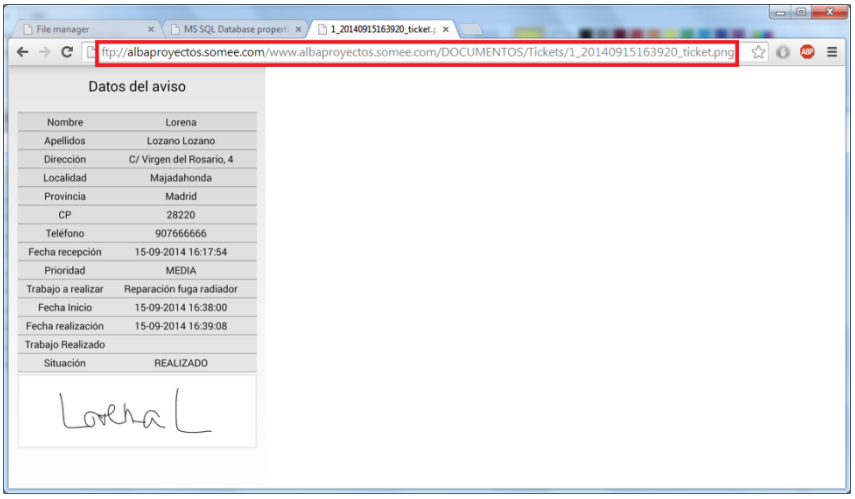


Figura 112. Pruebas modulares. Prueba de aplicación cliente .Net: Prueba de visualización de documentos. Visualización de documento del repositorio Web desde el navegador.

A continuación se comprueba la visualización del ticket desde las tres posibles vías de acceso:

- Detalle de avisos en la rejilla Clientes-Avisos-Documentos (id aviso = 1).

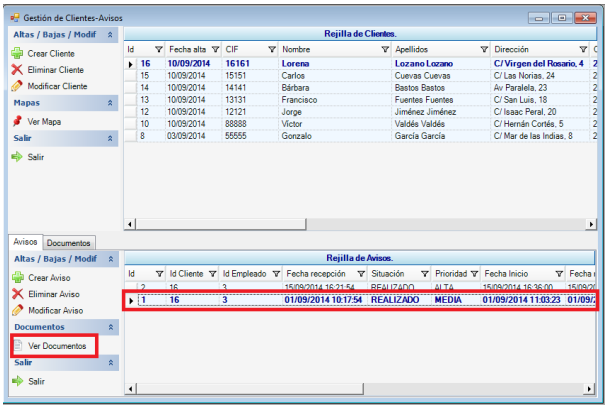


Figura 113. Pruebas modulares. Prueba de aplicación cliente .Net: Prueba de visualización de documentos. Documentos asociados a aviso.

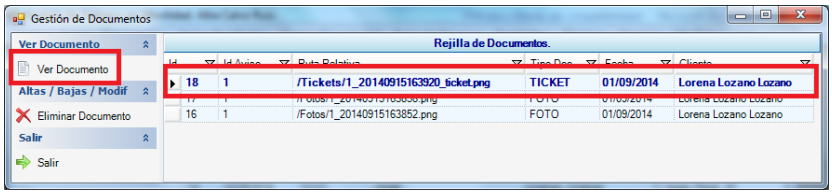


Figura 114. Pruebas modulares. Prueba de aplicación cliente .Net: Prueba de visualización de documentos. Visualización de documento desde la aplicación cliente .Net.

- Detalle de documentos en la rejilla Clientes-Avisos-Documentos (id cliente = 16).

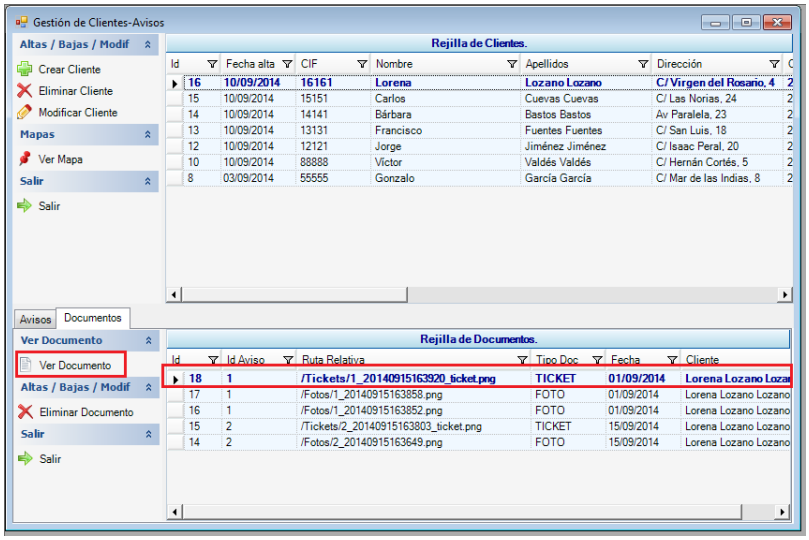


Figura 115. Pruebas modulares. Prueba de aplicación cliente .Net: Prueba de visualización de documentos. Documentos asociados a cliente.

- Rejilla general de gestión documental (filtrando por id aviso 1, tipo de ticket y fecha 15/09/2014).

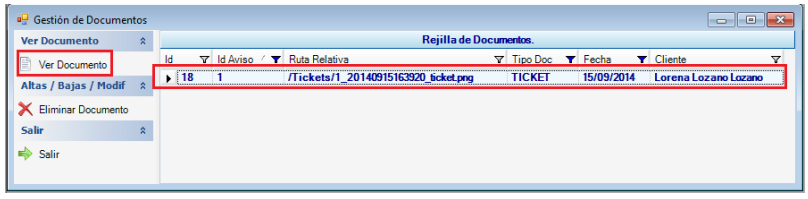


Figura 116. Pruebas modulares. Prueba de aplicación cliente .Net: Prueba de visualización de documentos. Visualización de documento desde la aplicación cliente .Net 2.

El resultado de todas las opciones es la presentación del siguiente formulario con la visualización del documento:

Figura 117. Pruebas modulares. Prueba de aplicación cliente .Net: Prueba de visualización de documentos. Documento justificante firmado por el cliente.

5.1.2 Pruebas API Web

5.1.2.1 Diseño de la prueba

La prueba consiste en la realización de solicitudes HTTP GET y POST a la API Web utilizando el plugin del navegador Google Chrome "Advanced Rest Client".

5.1.2.2 Análisis de resultados

A continuación se plasman los dos ejemplos básicos de solicitud a la API Rest:

- Obtención de todos los avisos de la aplicación.

GET http://www.albaproyectos.somee.com/SLIM_REST_API/restApi.php/avisos

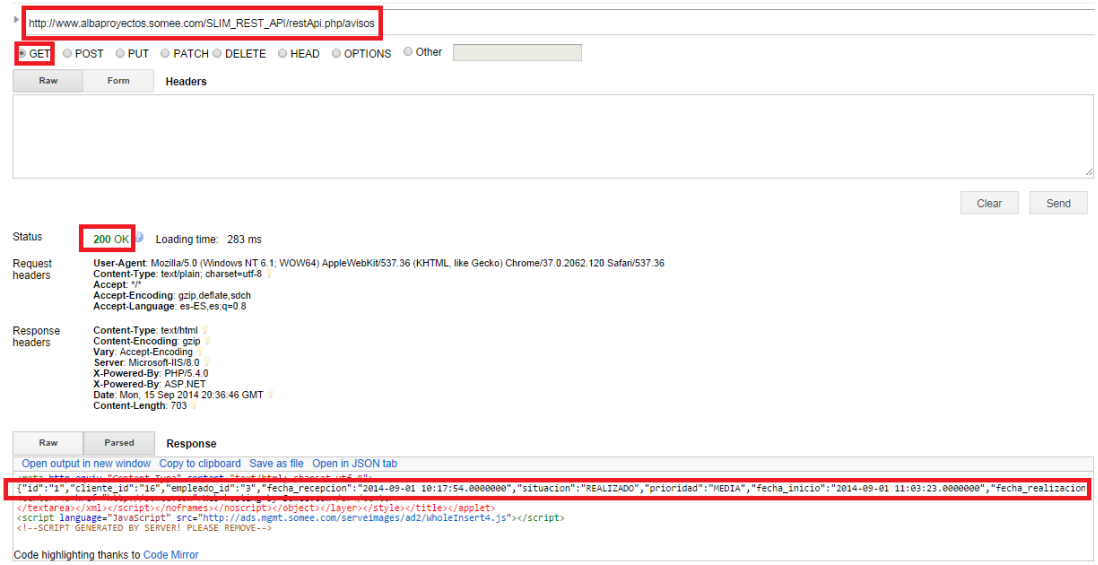


Figura 118. Pruebas modulares. Prueba de aplicación API Web: Prueba de consulta a la base de datos y obtención de objetos JSON por el método GET.

En la imagen se muestra la URL, el método utilizado (GET), el código de respuesta del servidor (200 OK) y, por último, la lista de avisos en formato JSON en el cuerpo de la página.

- Inserción en base de datos de una coordenada de tracking mediante el envío por método POST de un objeto JSON con los detalles de la misma (empleado_id, latitud, longitud, fecha y repeticiones).

POST http://www.albaproyectos.somee.com/SLIM_REST_API/restApi.php/tracking/3
JSON { "empleado_id" : "3", "latitud" : "-3.867290", "longitud" : "40.474426", "fecha" : "2014-09-15 22:57:34", "repeticiones" : "5" }

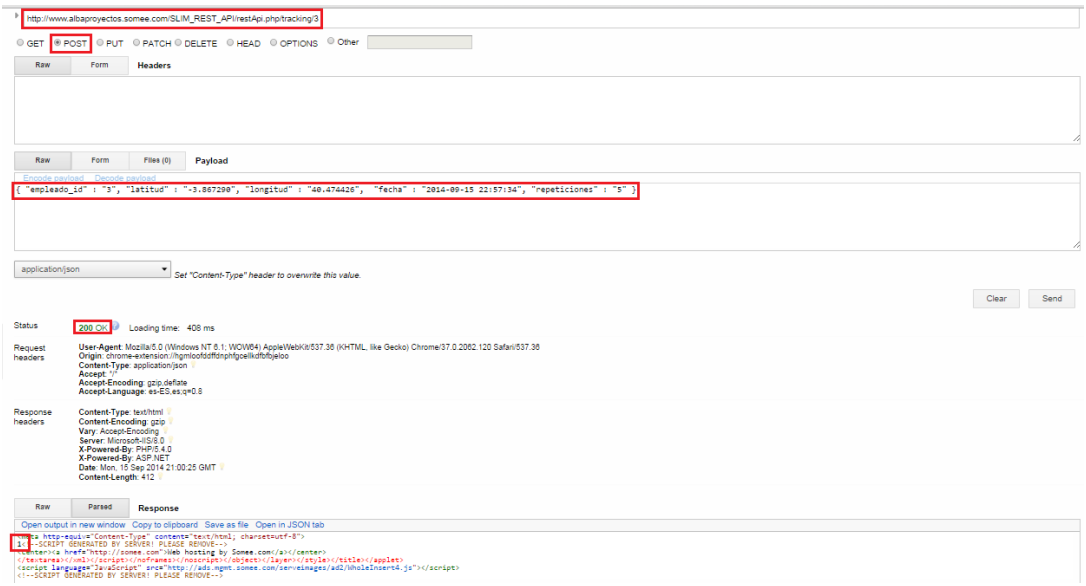


Figura 119. Pruebas modulares. Prueba de aplicación API Web: Prueba de inserción en base de datos mediante el envío de un objeto JSON por el método POST.

En la imagen se muestra la URL, el método (POST), el objeto JSON enviado con los datos de la coordenada, el código de respuesta del servidor (200 OK) y el número de filas insertadas en el cuerpo de la página.

Se puede comprobar la correcta inserción de la coordenada de tracking realizando una consulta a la base de datos:

```
select * from tracking
```

100 %

Resultados | Mensajes

	id	empleado_id	longitud	latitud	fecha	repeticiones
1	1	3	-3.856903	40.474051	2014-09-10 09:56:47.0000000	2
2	2	3	-3.859302	40.473308	2014-09-10 09:58:13.0000000	0
3	3	3	-3.860276	40.472617	2014-09-10 09:59:28.0000000	1
4	4	3	-3.862010	40.472166	2014-09-10 10:01:01.0000000	0
5	5	3	-3.861940	40.470789	2014-09-10 10:02:46.0000000	0
6	6	3	-3.862567	40.469591	2014-09-10 10:04:05.0000000	25
7	7	3	-3.862548	40.468399	2014-09-10 10:30:59.0000000	0
8	8	3	-3.863881	40.467447	2014-09-10 10:32:43.0000000	0
9	9	3	-3.863425	40.466372	2014-09-10 10:35:35.0000000	1
10	10	3	-3.865881	40.465660	2014-09-10 10:36:53.0000000	0
11	11	3	-3.866643	40.464277	2014-09-10 10:38:34.0000000	0
12	12	3	-3.867169	40.466566	2014-09-10 10:39:21.0000000	0
13	13	3	-3.867315	40.468735	2014-09-10 10:40:47.0000000	18
14	14	3	-3.867937	40.471061	2014-09-10 10:59:12.0000000	0
15	15	3	-3.869151	40.472599	2014-09-10 11:01:58.0000000	0
16	16	3	-3.869826	40.472770	2014-09-10 11:03:39.0000000	33
17	17	3	-3.872633	40.473936	2014-09-10 11:36:23.0000000	2
18	18	3	40.474426	-3.867290	2014-09-15 22:57:34.0000000	5

Figura 120. Pruebas modulares. Prueba de aplicación API Web: Prueba de correcta inserción en base de datos de la coordenada enviada en forma de objeto JSON en el paso anterior.

5.1.3 Pruebas de aplicación móvil de gestión de avisos y tracking GPS

Para realizar esta prueba se necesita un entorno con las siguientes características:

- Smartphone con sistema operativo Android (pruebas realizadas con versión 4.4.4).
- Entorno de desarrollo Eclipse.

5.1.3.1 Pruebas realizadas

5.1.3.1.1 Pruebas de realización de login contra la API Web

5.1.3.1.1.1. Diseño de la prueba

La prueba consiste en la comprobación del correcto envío de las credenciales de usuario a la API Web y el acceso a la aplicación cuando éstas son correctas.

5.1.3.1.1.2. Análisis de los resultados

Se ha comprobado que la tarea asíncrona encargada de la realización del login accede correctamente a la API Web y permite el acceso sólo a los usuarios con rol de técnico que hayan proporcionado una credenciales válidas.

El resultado ha sido el esperado: en caso de nick o contraseña incorrectos o de usuario no autorizado, se muestra un mensaje de informe de error:

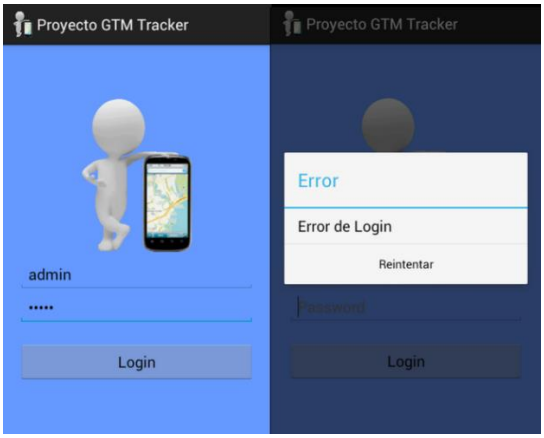


Figura 121. Pruebas modulares. Prueba de aplicación cliente Android: Prueba de autenticación contra la API Web.

En caso de credenciales válidas, se accede al menú de la aplicación:

Figura 122. Pruebas modulares. Prueba de aplicación cliente Android: Menú de la aplicación.

5.1.3.1.2 Pruebas de servicio de tracking GPS

5.1.3.1.2.1. Diseño de la prueba

La prueba consiste en la comprobación de la correcta ejecución del servicio, de su permanencia aunque la aplicación principal se cierre y del correcto almacenamiento de las coordenadas registradas en la base de datos mediante la utilización de la API Web.

5.1.3.1.2.2. Análisis de resultados

Se ha comprobado la correcta iniciación y detención del servicio al pulsar los botones correspondientes mediante la aparición de un icono que simboliza el tracking en la barra superior de la aplicación:

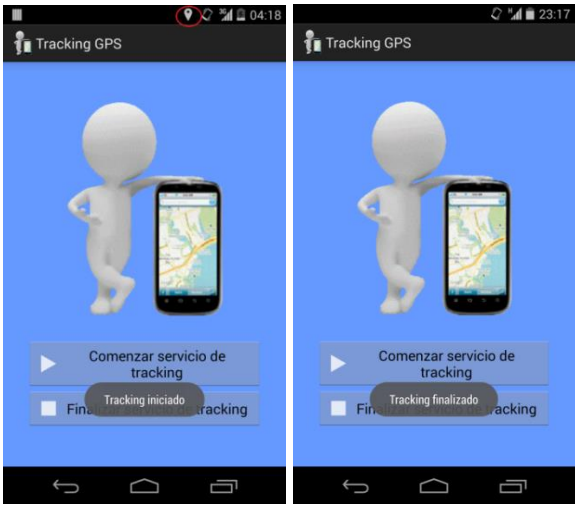


Figura 123. Pruebas modulares. Prueba de aplicación cliente Android: Prueba de inicio y detención del servicio de tracking GPS.

Además, se ha comprobado la persistencia del servicio al pasar la aplicación a segundo plano o ser cerrada por el planificador de procesos.

Por último, se ha comprobado el correcto almacenamiento en la bases de datos de las coordenadas mediante la realización de rutas poniéndose en el lugar de un técnico. El resultado de una de ellas en base de datos se muestra en la imagen inferior, donde las repeticiones mayores de 5 representan las paradas del técnico:

	id	empleado_id	longitud	latitud	fecha	repeticiones
1	1	3	-3.856903	40.474051	2014-09-10 09:56:47.0000000	2
2	2	3	-3.859302	40.473308	2014-09-10 09:58:13.0000000	0
3	3	3	-3.860276	40.472617	2014-09-10 09:59:28.0000000	1
4	4	3	-3.862010	40.472166	2014-09-10 10:01:01.0000000	0
5	5	3	-3.861940	40.470789	2014-09-10 10:02:46.0000000	0
6	6	3	-3,862567	40,469591	2014-09-10 10:04:05.0000000	25
7	7	3	-3.862548	40.468399	2014-09-10 10:30:59.0000000	0
8	8	3	-3.863881	40.467447	2014-09-10 10:32:43.0000000	0
9	9	3	-3.863425	40.466372	2014-09-10 10:35:35.0000000	1
10	10	3	-3.865881	40.465660	2014-09-10 10:36:53.0000000	0
11	11	3	-3.866643	40.464277	2014-09-10 10:38:34.0000000	0
12	12	3	-3.867169	40.466566	2014-09-10 10:39:21.0000000	0
13	13	3	-3.867315	40.468735	2014-09-10 10:40:47.0000000	18
14	14	3	-3.867937	40.471061	2014-09-10 10:59:12.0000000	0
15	15	3	-3.869151	40.472599	2014-09-10 11:01:58.0000000	0
16	16	3	-3.869826	40,472770	2014-09-10 11:03:39.0000000	33
17	17	3	-3.872633	40.473936	2014-09-10 11:36:23.0000000	2

Figura 124. Pruebas modulares. Prueba de aplicación cliente Android: Consulta de las coordenadas almacenadas en base de datos por el servicio de tracking GPS.

5.1.3.1.3 Pruebas de gestión de avisos

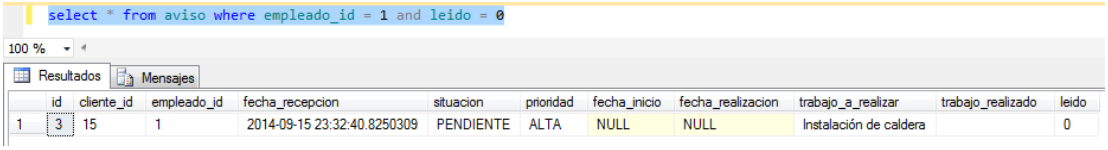
5.1.3.1.2.1. Diseño de la prueba

La prueba consiste en la comprobación de la correcta recepción, almacenamiento y consulta de avisos desde la aplicación

5.1.3.1.2.2. Análisis de resultados

- Recepción de avisos

  Realizando una consulta a la base de datos se pueden ver los avisos asociados al técnico Juan (id 1):



	id	cliente_id	empleado_id	fecha_recepcion	situacion	prioridad	fecha_inicio	fecha_realizacion	trabajo_a_realizar	trabajo_realizado	leido
1	3	15	1	2014-09-15 23:32:40.8250309	PENDIENTE	ALTA	NULL	NULL	Instalación de caldera		0

Figura 125. Pruebas modulares. Prueba de aplicación cliente Android: Prueba de gestión de avisos. Consulta de avisos asociados a técnico.

Este aviso debe figurar en la pantalla de nuevos avisos de la aplicación:

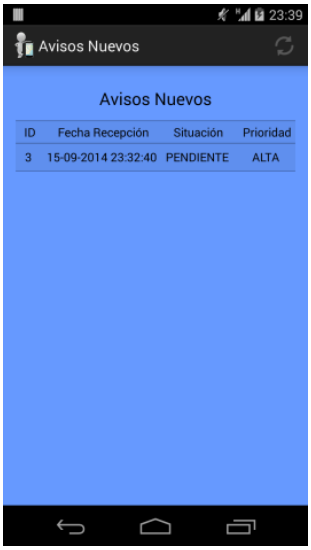


Figura 126. Pruebas modulares. Prueba de aplicación cliente Android: Prueba de gestión de avisos. Menú de avisos nuevos.

Además, una vez el técnico lo marque como leído, esto debe reflejarse en la base de datos tras la sincronización:

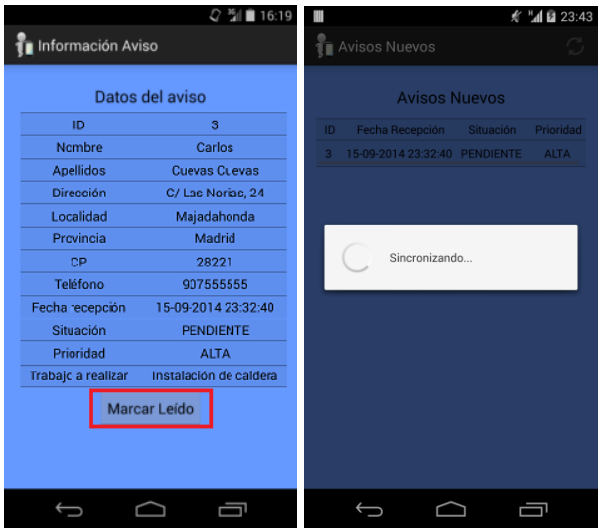


Figura 127. Pruebas modulares. Prueba de aplicación cliente Android: Prueba de gestión de avisos. Prueba de marcado de aviso como leído en la aplicación cliente Android.

```
select * from aviso where empleado_id = 1 and leido = 1
```

100 %

Resultados | Mensajes

	id	cliente_id	empleado_id	fecha_recepcion	situacion	prioridad	fecha_inicio	fecha_realizacion	trabajo_a_realizar	trabajo_realizado	leido
1	3	15	1	2014-09-15 23:32:40.0000000	PENDIENTE	ALTA	NULL	NULL	Instalación de caldera		1

Figura 128. Pruebas modulares. Prueba de aplicación cliente Android: Prueba de gestión de avisos. Prueba de marcado de aviso como leído mediante consulta a base de datos.

- Consulta y cumplimentación de avisos pendientes

Los avisos pendientes del técnico tienen que poder ser consultados desde la aplicación:

```
select * from aviso where empleado_id = 1 and leido = 1 and situacion = 'PENDIENTE'
```

100 %

Resultados | Mensajes

	id	cliente_id	empleado_id	fecha_recepcion	situacion	prioridad	fecha_inicio	fecha_realizacion	trabajo_a_realizar	trabajo_realizado	leido
1	3	15	1	2014-09-15 23:32:40.8250309	PENDIENTE	ALTA	NULL	NULL	InstalaciÃ³n de caldera		1

Figura 129. Pruebas modulares. Prueba de aplicación cliente Android: Prueba de gestión de avisos. Prueba de cumplimentación de aviso. Consulta inicial a la base de datos.

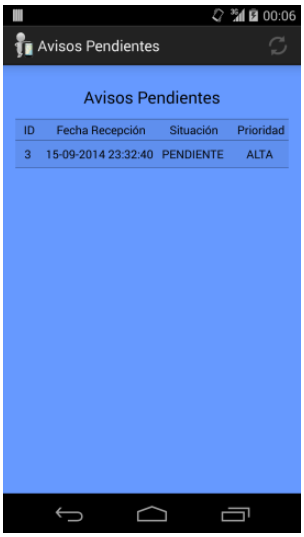


Figura 130. Pruebas modulares. Prueba de aplicación cliente Android: Prueba de gestión de avisos. Prueba cumplimentación de aviso en la aplicación cliente Android.

Al cumplimentar el aviso introduciendo documentos asociados (fotos y ticket firmado por el cliente), los documentos deben almacenarse en el repositorio de ficheros y en BD local y sincronizarse con la base de datos y repositorio de ficheros Web cuando el técnico disponga de conectividad:

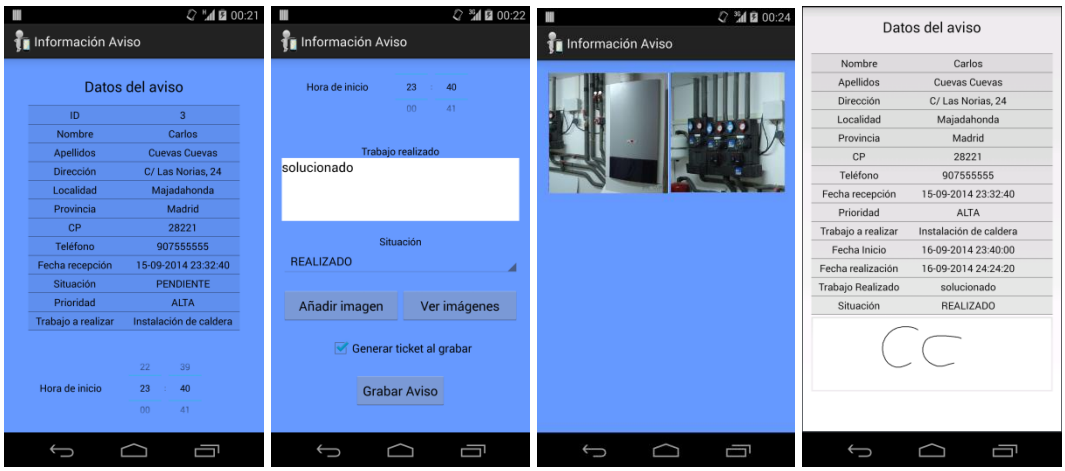


Figura 131. Pruebas modulares. Prueba de aplicación cliente Android: Prueba de gestión de avisos. Prueba de cumplimentación de aviso. Introducción de datos relativos al aviso (hora de inicio, trabajo realizado, estado), adición de fotos y generación de ticket firmado por el cliente.

Estos documentos se encuentran en el repositorio de ficheros local del dispositivo:

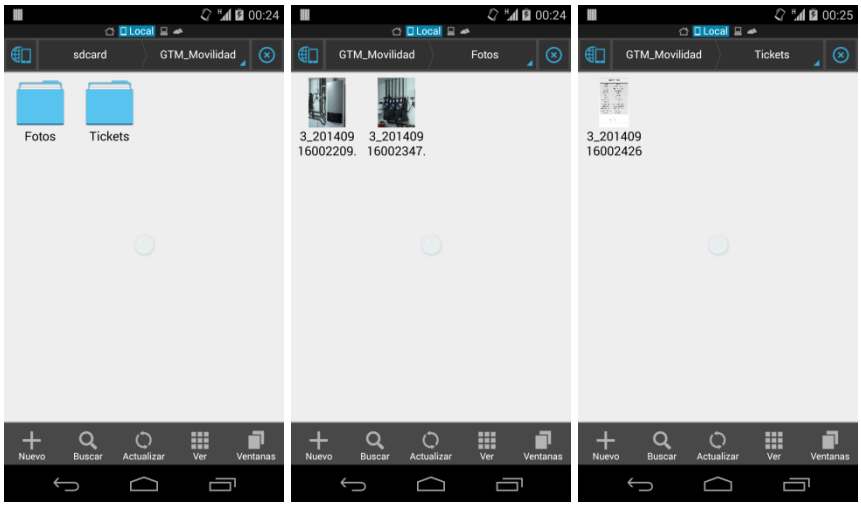


Figura 132. Pruebas modulares. Prueba de aplicación cliente Android: Prueba de gestión de avisos. Almacenamiento de fotos y ticket firmado por el cliente asociados a un aviso en el repositorio de ficheros local del teléfono.

Al realizar la sincronización, estos ficheros pasan a estar disponibles en el repositorio Web y, además, su ruta relativa y aviso asociado se almacenan en la base de datos Web:

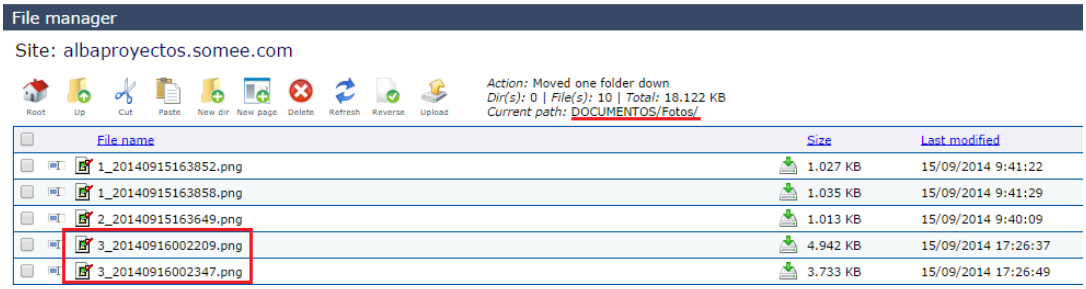


Figura 133. Pruebas modulares. Prueba de aplicación cliente Android: Prueba de gestión de avisos. Almacenamiento de fotos asociadas a un aviso en el repositorio Web

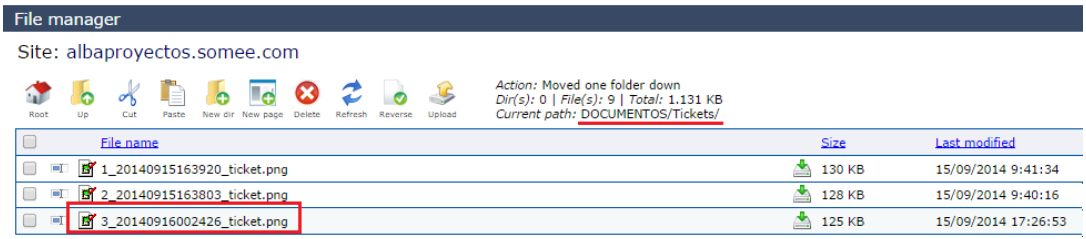
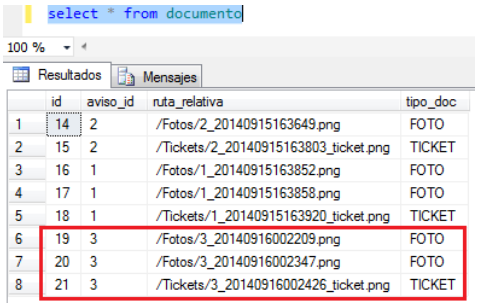


Figura 134. Pruebas modulares. Prueba de aplicación cliente Android: Prueba de gestión de avisos. Almacenamiento de ticket generado para un aviso en el repositorio Web.



	id	aviso_id	ruta_relativa	tipo_doc
1	14	2	/Fotos/2_20140915163649.png	FOTO
2	15	2	/Tickets/2_20140915163803_ticket.png	TICKET
3	16	1	/Fotos/1_20140915163852.png	FOTO
4	17	1	/Fotos/1_20140915163858.png	FOTO
5	18	1	/Tickets/1_20140915163920_ticket.png	TICKET
6	19	3	/Fotos/3_20140916002209.png	FOTO
7	20	3	/Fotos/3_20140916002347.png	FOTO
8	21	3	/Tickets/3_20140916002426_ticket.png	TICKET

Figura 135. Pruebas modulares. Prueba de aplicación cliente Android: Prueba de gestión de avisos. Almacenamiento de fotos y ticket en la base de datos Web.

- Consulta de historial de avisos realizados el último mes

  Prueba de consulta de avisos realizados con visualización de sus documentos asociados almacenados en el repositorio local:

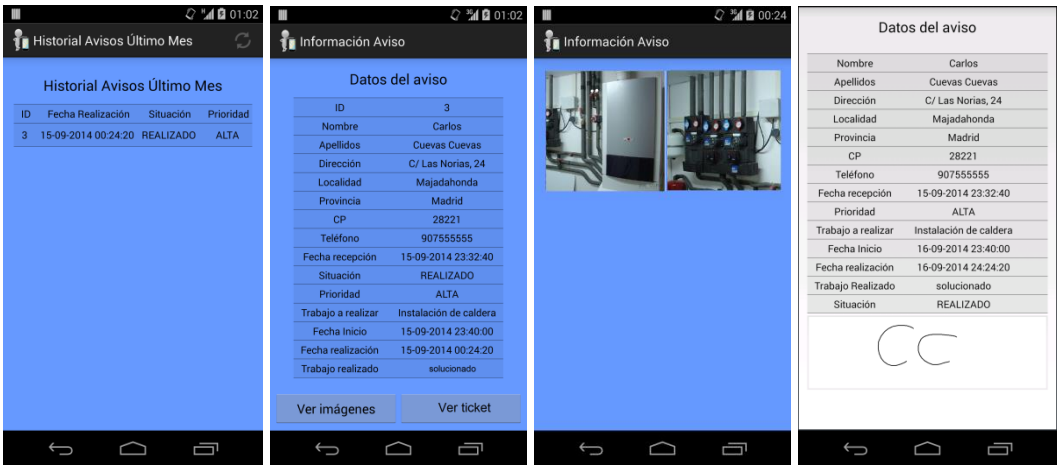


Figura 136. Pruebas modulares. Prueba de aplicación cliente Android: Prueba de gestión de avisos. Consulta de historial de avisos realizados el último mes, incluyendo fotos y tickets firmados por el cliente.

5.2 Pruebas de integración

5.2.1 Diseño de la prueba

Tras probar cada uno de los módulos por separado se ha procedido a comprobar que la interactuación entre ellos era correcta. Por ejemplo: los avisos asociados a un técnico dados de alta en la aplicación en C# deben aparecer en la aplicación móvil, las coordenadas registradas en la aplicación Android deben poder ser trasladadas a un mapa MapPoint y presentadas al usuario en la aplicación cliente en C#, etc.

5.2.2 Análisis de resultados

Todos los módulos probados previamente acceden la misma base de datos Web y realizan cambios en ella.

Dado que todos ellos funcionaban correctamente por separado reflejando sus cambios en la misma, no ha habido ningún problema de integración.

6 Conclusiones y trabajo futuro

6.1 Conclusiones

La gestión de empresas con actividad de SAT y, más concretamente, la implantación de servicios de movilidad dentro de las mismas, se ha identificado como una necesidad real que el proyecto podría abordar.

En respuesta a esa necesidad, se ha desarrollado una solución de gestión de avisos en movilidad con los siguientes componentes:

- Aplicación cliente .Net de gestión de clientes y avisos y presentación de mapas de ubicación de los mismos y de rutas de técnicos.

- Aplicación cliente Android para la gestión de avisos y tracking GPS de terminales en movilidad.

La solución, tal y como se ha implementado, es utilizable como producto autónomo por empresas SAT, siendo sus prestaciones más destacadas:

- El envío inmediato de avisos a los técnicos y la grabación por parte de éstos del trabajo realizado en sus terminales, incluyendo subida de fotos y tickets (justificantes) firmados por el cliente.

- La representación cartográfica de ubicaciones y rutas de técnicos en la aplicación cliente manejada por los operadores de cara a la asignación eficiente de avisos y al control de absentismo.

- La gestión documental de fotos y justificantes de trabajo firmados por los clientes subidos desde los terminales en movilidad de los técnicos.

El producto soporta una funcionalidad genérica y cuenta con una arquitectura de fácil implantación. Esto debería permitir la realización de ofertas a un elevado número de empresas. Las ofertas podrían incluir la posibilidad de ampliación para cubrir funcionalidad adicional demandada por el cliente. En función de la realimentación recibida, el producto podría ir evolucionando.

En cuanto a la parte formativa, la elaboración del proyecto me ha permitido tomar contacto con las siguientes tecnologías:

- SQLServer y Management Studio 2012 para la gestión de bases de datos.

- Desarrollo de aplicaciones Windows Forms en C# utilizando controles de Infragistics en la capa de presentación y ORM Microsoft Entity Framework en la capa de acceso a datos.

- Representación cartográfica utilizando las librerías de MapPoint para .Net.

- Desarrollo de aplicaciones multihilo en Java para Android con soporte para el trabajo offline, utilizando una base de datos SQLite local y realizando

sincronización con la base de datos del servidor Web ante la presencia de conectividad.

- Gestión documental de fotos y justificantes de trabajo desde aplicación cliente .Net y Android.

- Desarrollo de una API Web con la tecnología REST y utilización del framework Slim para simplificar la tarea. Esta API da soporte a la interoperabilidad de la aplicación instalada en los terminales en movilidad con la base de datos.

- Creación de paquetes de instalación.

Técnicamente, el mayor reto ha sido la sincronización de la base de datos y el repositorio de ficheros local desde la aplicación de los terminales en movilidad con la base de datos y el repositorio de ficheros alojados en el servidor Web. La implementación de la solución utilizando el patrón SyncAdapter detallado en la sección "Desarrollo" no ha sido trivial y ha sido necesario consultar varias fuentes para resolver el problema.

Como conclusiones finales cabe destacar, por un lado, que el proyecto GTM me ha permitido una toma de contacto con el modelo de gestión de empresas SAT y con un variado conjunto de tecnologías necesarias para la implantación, y por otro, que el producto resultante es una solución conectada con el mundo real que podría constituir el embrión de un producto con un fuerte potencial de comercialización en el sector de empresas SAT.

6.2 Trabajo futuro

El proyecto podría ser objeto de mejora en relación con algunas limitaciones en su implementación actual:

- El proyecto da soporte a la gestión de avisos en movilidad de una empresa SAT. El alcance del proyecto no incluye aspectos de la actividad SAT como mantenimiento preventivo de instalaciones, revisiones periódicas, gestión de almacén, etc.

- La aplicación cliente de gestión presenta las limitaciones típicas de este tipo de solución en relación con la posibilidad de trabajar de manera deslocalizada y con la necesidad de realizar mantenimiento en el PC de cada usuario. No obstante, en pequeñas empresas estas limitaciones se atenúan y pierden peso frente a la ventaja de una mayor productividad en el desarrollo de la interfaz de usuario en el entorno de aplicación cliente.

En relación con las limitaciones anteriores y con el fin de conseguir un producto más competitivo, se podrían introducir mejoras en el proyecto en aspectos como:

- Ampliación de funcionalidad para dar un mayor soporte a la gestión de una empresa SAT, dando respuesta a necesidades como registro de instalaciones objeto de mantenimiento preventivo o correctivo, planificación automática de tareas periódicas, reparto automático de avisos recibidos, estadísticas, gestión de almacén, etc.

- Desarrollo de un módulo financiero (facturación, contabilidad, etc.) o integración con productos de terceros. Las facturas emitidas y otros documentos con o sin firma digital podrían integrarse en el módulo de gestión documental desarrollado en esta versión 2.0 de GTM.

- Extensión de la aplicación cliente para terminales en movilidad a otros sistemas operativos diferentes a Android.

- Integración del módulo de movilidad con aplicaciones sectoriales de terceros que no cuenten con servicio de movilidad, previo acuerdo con sus desarrolladores.

Aunque de algunas de las ampliaciones propuestas se derivan incertidumbres técnicas, existe una gran relación entre éstas y la implementación actual del proyecto GTM v2.0, de la cual son una evolución.

Gracias al bagaje adquirido durante la elaboración del proyecto, el desarrollo de las mejoras anteriores sería factible invirtiendo, claro está, el tiempo y esfuerzo necesarios.

Bibliografía

[1] Raúl Roses, Jaime Patricio Zarate Piray, "Estudio comparativo de las tecnologías Punto Net y Java para el desarrollo de aplicaciones Web"

[2] Gabriel Herraiz Antón, "ANDROID", Noviembre 2012, páginas 15-26

[3] Unnikrishnan.P.K, "Blackberry technology, a seminar report", School of Engineering COCHIN University Of Science & Technology, KOCHI-682022, August 2008

[4] Rafael Navarro Marset, "REST vs Web Services"

[5] Jack Cox, "SOAP vs. REST For Mobile Services", Septiembre 2011

[6] Santiago Gómez Ruiz, "Microsoft SQL Server, MySQL y PostgreSQL"

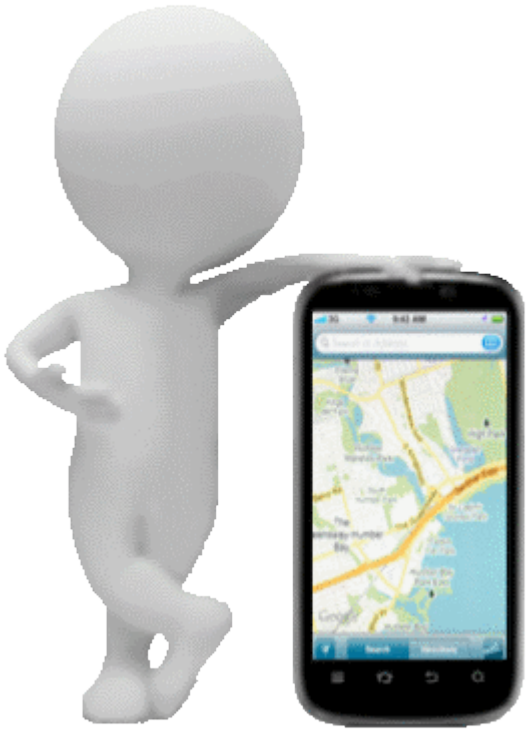
[7] Daniel Bartholome, "Comparamos MariaDB y MySQL cara a cara", Linux Magazine, número 73, páginas 44-46.

[8] Juan Ferrer Martínez, "Implantación de Aplicaciones Web", 2012, Capítulo 1

[9] José Barato, Israel Marcos, Javier Cantalapiedra, Carlos Von Prabucki, Prudencio Poza, Ignacio Pérez, "Plataformas de Arquitectura de Aplicaciones. El fin de la discusión J2EE VS .Net", Atos Origin, Madrid, 2006

Anexos

A Manual de instalación



ALBAPROYECTOS S.L.

MANUAL DE INSTALACIÓN

Gestión de Trabajos en Movilidad (GTM) v2.0

ÍNDICE

1 Introducción iii
2 Requisitos software iii
3 Requisitos de acceso a recursos del servidor iii
4 Posibles escenarios de instalación iii
    4.1 Casuística de escenarios iii
    4.2 Paquetes a instalar en cada caso. iv
5 Descripción de la instalación de paquetes iv
    5.1 Instalación en PCs iv
    5.2 Instalación en dispositivos móviles con Android v

1 Introducción

En los apartados siguientes se describen los pasos necesarios para la instalación de la aplicación Gestión de Trabajos en Movilidad (GTM) v2.0.

Para realizar la instalación debe disponer de los siguientes paquetes de instalación:

- InstalacionProyectoGTM_v2.exe
- GTM_Movilidad.apk

2 Requisitos software

Para que la aplicación funcione correctamente es necesario que cada terminal PC de trabajo cumpla los siguientes requisitos:

- Sistema operativo Windows 7.
- .Net Framework 4.0 o superior
- Microsoft MapPoint Europe 2013

Asimismo, es necesario que los terminales móviles dispongan de la versión de Android 4.4.4.

3 Requisitos de acceso a recursos del servidor

La aplicación accede a una base de datos en la nube accesible a través de un servidor de base de datos SQLServer 2012.

4 Posibles escenarios de instalación

4.1 Casuística de escenarios

Son posibles 3 escenarios de instalación:

1) PCs pertenecientes a un usuario con rol de “administrador”.
2) PCs pertenecientes a un usuario con rol de “empleado”.
3) Terminales móviles pertenecientes a un usuario con rol de “técnico”.

4.2 Paquetes a instalar en cada caso.

Caso nº	Tipo de Usuario	Paquetes a instalar
1	Administrador	InstalacionProyectoGTM_v2.exe
2	Empleado	InstalacionProyectoGTM_v2.exe
3	Técnico	GTM_Movilidad.apk

5 Descripción de la instalación de paquetes

5.1 Instalación en PCs

Ejecute el paquete de instalación “InstalacionProyectoGTM_v2.exe”:

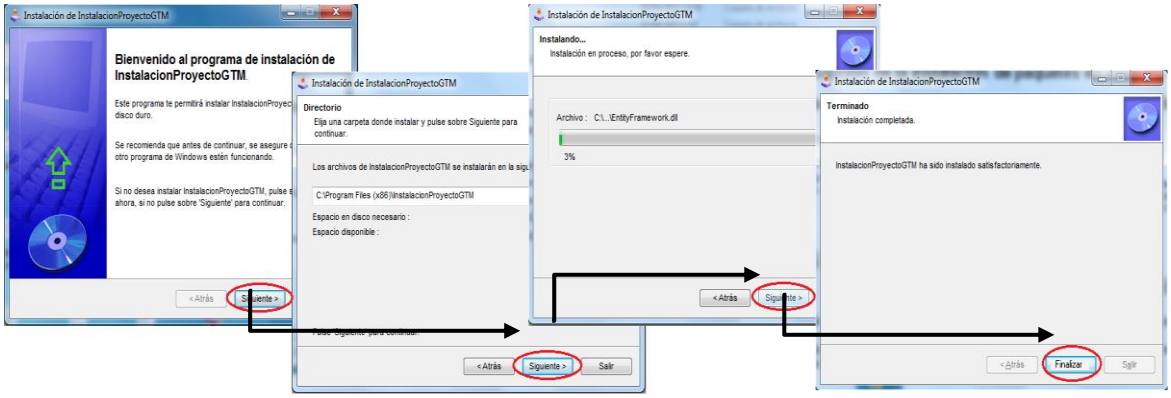


Figura 137. Anexo I: Manual de instalación. Paquete de instalación de la aplicación cliente de gestión.

Una vez finalizada la instalación, aparecerán dos accesos directos a la aplicación en el escritorio y en el menú de inicio:

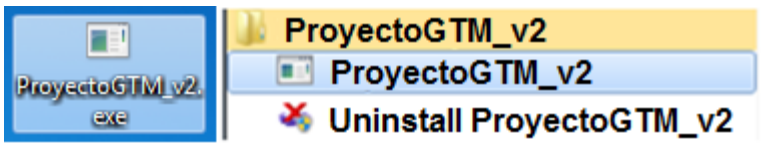


Figura 138. Anexo I: Manual de instalación. Accesos directos de la aplicación cliente de gestión.

5.2 Instalación en dispositivos móviles con Android

Copie el fichero “GTM_Movilidad.apk” a la memoria interna del teléfono y selecciónelo para llevar a cabo el proceso de instalación:

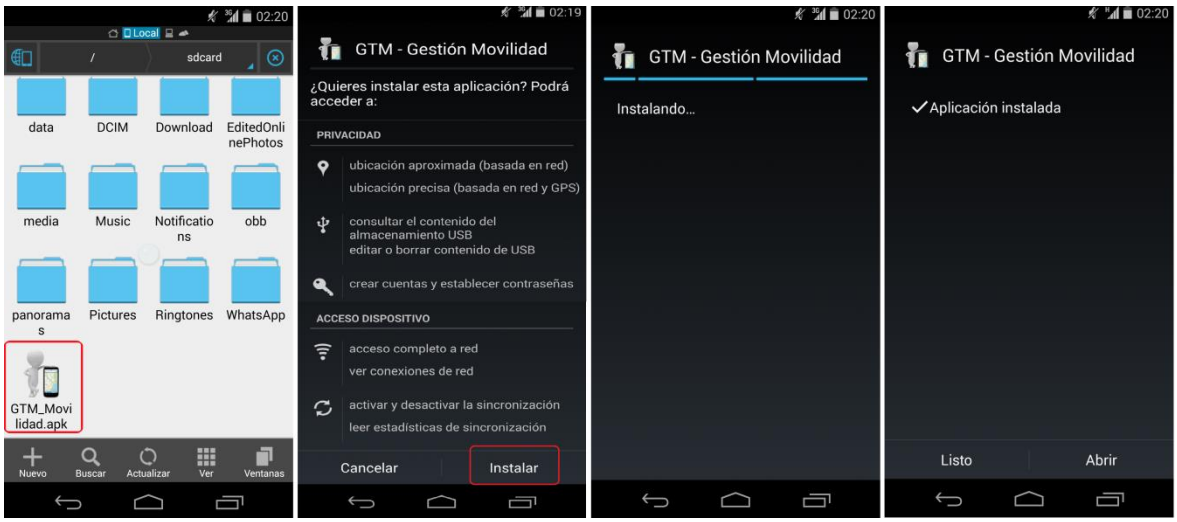


Figura 139. Anexo I: Manual de instalación. Instalación de la aplicación Android.